Model Based Imaging

Charles A. Bouman

Purdue University
To my family for their patient support
Contents

1 Introduction 1
   1.1 What is Model-Based Image Processing? ................. 2
   1.2 How can I learn Model-Based Image Processing? ........ 8

2 Probability, Estimation, and Random Processes 13
   2.1 Random Variables and Expectation .................. 13
   2.2 Some Commonly Used Distributions .................. 18
   2.3 Frequentist and Bayesian Estimators ................ 23
      2.3.1 Frequentist Estimation and the ML Estimator .... 23
      2.3.2 Bayesian Estimation and the MAP Estimator .... 27
      2.3.3 Comparison of the ML and MMSE Estimators .... 34
   2.4 Discrete-Time Random Processes .................... 35
   2.5 Chapter Problems .................................... 41

3 Causal Gaussian Models 45
   3.1 Causal Prediction in Gaussian Models ............... 45
   3.2 Density Functions Based on Causal Prediction ........ 48
   3.3 1-D Gaussian Autoregressive (AR) Models ........... 49
   3.4 2-D Gaussian AR Models ............................. 54
   3.5 Chapter Problems .................................... 59

4 Non-Causal Gaussian Models 65
   4.1 Non-Causal Prediction in Gaussian Models ........... 66
   4.2 Density Functions Based on Non-Causal Prediction .... 67
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>1-D Gaussian Markov Random Fields (GMRF)</td>
<td>69</td>
</tr>
<tr>
<td>4.4</td>
<td>2-D Gaussian Markov Random Fields (GMRF)</td>
<td>72</td>
</tr>
<tr>
<td>4.5</td>
<td>Relation Between GMRF and Gaussian AR Models</td>
<td>73</td>
</tr>
<tr>
<td>4.6</td>
<td>GMRF Models on General Lattices</td>
<td>77</td>
</tr>
<tr>
<td>4.7</td>
<td>Chapter Problems</td>
<td>80</td>
</tr>
<tr>
<td>5</td>
<td>MAP Estimation with Gaussian Priors</td>
<td>83</td>
</tr>
<tr>
<td>5.1</td>
<td>MAP Image Restoration</td>
<td>84</td>
</tr>
<tr>
<td>5.2</td>
<td>Computing the MAP Estimate</td>
<td>88</td>
</tr>
<tr>
<td>5.3</td>
<td>Gradient Descent Optimization</td>
<td>89</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Convergence Analysis of Gradient Descent</td>
<td>90</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Frequency Analysis of Gradient Descent</td>
<td>93</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Gradient Descent with Line Search</td>
<td>96</td>
</tr>
<tr>
<td>5.4</td>
<td>Iterative Coordinate Descent Optimization</td>
<td>97</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Convergence Analysis of Iterative Coordinate Descent</td>
<td>100</td>
</tr>
<tr>
<td>5.5</td>
<td>Conjugate Gradient Optimization</td>
<td>103</td>
</tr>
<tr>
<td>5.6</td>
<td>Chapter Problems</td>
<td>105</td>
</tr>
<tr>
<td>6</td>
<td>Non-Gaussian MRF Models</td>
<td>109</td>
</tr>
<tr>
<td>6.1</td>
<td>Continuous MRFs Based on Pair-Wise Cliques</td>
<td>109</td>
</tr>
<tr>
<td>6.2</td>
<td>Selection of Potential and Influence Functions</td>
<td>114</td>
</tr>
<tr>
<td>6.3</td>
<td>Convex Potential Functions</td>
<td>117</td>
</tr>
<tr>
<td>6.4</td>
<td>Scaling Parameters for Non-Gaussian MRFs</td>
<td>123</td>
</tr>
<tr>
<td>6.5</td>
<td>Chapter Problems</td>
<td>127</td>
</tr>
<tr>
<td>7</td>
<td>MAP Estimation with Non-Gaussian Priors</td>
<td>129</td>
</tr>
<tr>
<td>7.1</td>
<td>The MAP Cost Function and Its Properties</td>
<td>129</td>
</tr>
<tr>
<td>7.2</td>
<td>General Approaches to Optimization</td>
<td>133</td>
</tr>
<tr>
<td>7.3</td>
<td>Optimization of the MAP Cost Function</td>
<td>138</td>
</tr>
<tr>
<td>7.4</td>
<td>Chapter Problems</td>
<td>143</td>
</tr>
</tbody>
</table>
8 Surrogate Functions and Majorization

8.1 Motivation and Intuition ........................................... 148
8.2 Formal Definition and Properties ................................. 149
8.3 Minimizing the MAP Cost Function .............................. 152
8.4 Application to Line Search ....................................... 159
8.5 Chapter Problems .................................................. 163

9 Constrained Optimization

9.1 Motivation and Intuition ........................................... 165
9.2 The Lagrangian and Dual Function .............................. 170
  9.2.1 Strong Duality .................................................. 173
  9.2.2 Technical Conditions for Strong Duality ................. 174
9.3 The Augmented Lagrangian ...................................... 177
9.4 Shrinkage and Proximal Maps .................................. 183
9.5 Variable Splitting and the ADMM Algorithm ................. 188
  9.5.1 Total Variation Regularization using ADMM ............ 191
  9.5.2 Enforcing Convex Constraints using ADMM ............. 193
  9.5.3 Convergence of ADMM Algorithm ......................... 194
9.6 Chapter Problems .................................................. 198

10 Plug-and-Play and Equilibrium Methods

10.1 Motivation and Intuition ........................................... 202
  10.1.1 The Plug-and-Play Algorithm and Agents ............... 204
  10.1.2 The Design of Plug-In Agents .............................. 206
  10.1.3 Are All Agents Proximal Maps? ............................ 209
10.2 Consensus Equilibrium for Two Models ....................... 210
  10.2.1 Solving the CE Equations .................................. 212
10.3 Multi-Agent Consensus Equilibrium ........................... 214
10.4 Chapter Problems .................................................. 219

11 Model Parameter Estimation .......................................... 221
# CONTENTS

11.1 Parameter Estimation Framework ............................................ 221
11.2 Noise Variance Estimation .................................................. 223
11.3 Scale Parameter Selection and Estimation .............................. 226
11.4 Order Identification and Model Selection .............................. 230
11.5 Chapter Problems .................................................................. 231

12 The Expectation-Maximization (EM) Algorithm .......................... 233
12.1 Motivation and Intuition ...................................................... 234
12.2 Gaussian Mixture Distributions ............................................. 237
12.3 Theoretical Foundation of the EM Algorithm ......................... 240
12.4 EM for Gaussian Mixtures .................................................... 242
12.5 Algorithmic Implementation of EM Clustering ....................... 244
12.6 EM Convergence and Majorization ...................................... 246
12.7 Simplified Methods for Deriving EM Updates ......................... 248
   12.7.1 Exponential Distributions and Sufficient Statistics ......... 249
   12.7.2 EM for Exponential Distributions ................................. 254
   12.7.3 EM for Multivariate Gaussian Mixtures ....................... 255
12.8 Cluster Order Selection ...................................................... 257
12.9 Chapter Problems .................................................................. 258

13 Markov Chains and Hidden Markov Models ................................. 263
13.1 Markov Chains ................................................................. 263
13.2 Parameter Estimation for Markov Chains .............................. 266
13.3 Hidden Markov Models ...................................................... 268
   13.3.1 State Sequence Estimation and Dynamic Programming ........ 269
   13.3.2 State Probability and the Forward-Backward Algorithm .... 271
   13.3.3 Training HMMs with the EM Algorithm ......................... 273
13.4 Stationary Distributions of Markov Chains ............................ 276
13.5 Properties of Markov Chains .............................................. 281
13.6 Chapter Problems .................................................................. 287
14 General MRF Models 291
14.1 MRFs and Gibbs Distributions .............................................. 292
14.2 1-D MRFs and Markov Chains ............................................ 297
14.3 Discrete MRFs and the Ising Model ...................................... 301
14.4 2-D Markov Chains and MRFs ............................................. 312
14.5 MRF Parameter Estimation .................................................. 316
14.6 Chapter Problems ............................................................. 321

15 Stochastic Simulation 323
15.1 Motivation for Simulation Methods ...................................... 323
15.2 The Metropolis Sampler ..................................................... 325
15.3 The Hastings-Metropolis Sampler ........................................ 330
15.4 Stochastic Sampling of MRFs .............................................. 335
15.5 The Gibbs Sampler ........................................................... 339
15.6 Chapter Problems ............................................................. 343

16 Bayesian Segmentation 347
16.1 Framework for Bayesian Segmentation ................................. 347
16.1.1 Discrete Optimization for MAP Segmentation .................. 351
16.1.2 Stochastic Integration for MPM Segmentation ................... 358
16.1.3 Multiscale Approaches to MAP Segmentation .................... 360
16.2 Joint Segmentation and Reconstruction ............................... 364
16.3 ML Parameter Estimation with Stochastic EM ...................... 369
16.4 Chapter Problems ............................................................. 377

17 Poisson Data Models 379
17.1 The Poisson Forward Model ............................................... 380
17.1.1 Poisson Noise Approximations ....................................... 382
17.1.2 Poisson Models for Emission Tomography ...................... 384
17.2 Poisson Surrogate Functions for MAP Estimation ................. 386
17.3 ICD Optimization with Surrogate Functions ....................... 387
Chapter 1

Introduction

With a quick glance of the front cover, anyone with normal vision can tell you what has happened. Something has fallen into a liquid, perhaps water, and caused a very symmetric and beautiful splash.

What seems like such a simple and natural response hides an enormously sophisticated process that is able to infer from a 2-D light field a very complex series of embedded phenomena. From the 2-D light field, we infer the 3-D scene structure; from the scene structure, we infer the context of a fluid flowing through space and propagating as a wave; and from the dynamic motion, we infer a cause, presumably a falling object of some kind that created a splash.

In fact, no one fully understands how humans solve such complex problems, and we are unlikely to have complete answers in our lifetime, but the example makes some things clear. First, such inverse problems represent a fundamentally unsolved class of problems that are of enormous importance. Second, humans can only solve these problems by using strong embedded knowledge of the physical world and the constraints it imposes on physical observations.

The following chapter lays out the philosophy that motivates the creation of model-based image processing as a discipline, and it explains how the structure and content of this book can help to teach the basic tools needed to practically solve this important class of problems.
1.1 What is Model-Based Image Processing?

The goal of this book is to introduce the basic analytical tools for solving a broad range of inverse problems using the methods of **model-based image processing**. Model-based image processing is a collection of techniques that have emerged over the past few decades that provide a systematic framework for the solution of inverse problems that arise in imaging applications.

![Figure 1.1: Many imaging problems require the solution of an inverse problem in which the objective is to recover an unknown image \( X \) from measurements \( Y \). Often there are additional unknown “nuisance” parameters denoted by \( \phi \) and \( \theta \) that determine characteristics of the physical system and the regularity conditions, respectively.](image)

Figure 1.1 illustrates the general form of most inverse problems. A physical system of some type provides measurements, \( Y \), that depend on an unknown signal or image, \( X \). The objective is then to determine the unknown signal or image, \( X \), from these measurements. Since \( X \) is not directly observed, this problem of determining \( X \) from \( Y \) is known as an inverse problem because it requires that the physical process that generated the measurements be inverted or reversed to create \( X \) from \( Y \).

Inverse problems occur often in real imaging systems. So for example, \( Y \) might represent the measurements of a volume \( X \) taken by an optical or electron microscope, or it might represent the voltages read-outs in the CMOS sensor of a cell-phone camera. Alternatively, \( Y \) might represent the measurements from a radio telescope of unknown astronomical structures, or the photon counts in a medical PET scanner. All these imaging systems, and many more, share this same unifying structure.

\[1\] A second kind of inverse problem often occurs in which it is necessary to determine the input to a system that produces a desired output.
1.1 What is Model-Based Image Processing?

Any algorithm used for computing the solution to an inverse problem will typically share certain common structures or components. So for example, the purpose of any inversion algorithm is to produce an estimate, $\hat{X}$, of the unknown image, $X$, from the observations, $Y$. Often, there are also unknown nuisance parameters of the system that we denote by $\phi$. These parameters can represent unknown calibration parameters, such as focus or noise gain, that are typically not of direct interest, but must be determined to solve the inversion problem. Other parameters, $\theta$, are needed to determine the amount of regularization or smoothing required in the inversion process.

The model-based approach is built on a foundation of probability, so both the physical system and image to be discovered, $X$, are modeled as random quantities. The so-called forward model of the system is embodied by $p(y|x)$, the conditional distribution of the data, $Y$, given the unknown image, $X$. The prior model of the unknown image is embodied by the assumed prior distribution, $p(x)$. Intuitively, $p(y|x)$ tells us everything about how the observations are related to the unknown. This includes both the deterministic properties of the imaging system, such as its geometry and sensitivity, as well as probabilistic properties such as the noise amplitude.

Figure 1.2 illustrates the typical approach to model-based inversion. The key idea is to first create a forward model of the physical imaging system, and then search for an input, $\hat{X}$, to the forward model that best matches the actual measurements. This is often a counter-intuitive approach since we start the process of computing an inverse by computing the forward model, rather than directly looking for an inverse to the measurements. However, an advantage of this approach is that we can in principle invert any system that we can model. So the same approach is applicable to nonlinear systems or other systems for which there is no obvious analytical inverse.

Another critically important ingredient to model-based inversion is the use of a prior model to characterize the likely behavior of real images. Our goal will be to compute a final estimate, $\hat{X}$, that represents a balance between fitting the system model, $p(y|x)$, and remaining consistent with the expectations of the prior model, $p(x)$. In this text, we will primarily use the tools of Bayesian inference as a general framework for balancing these two objectives. While other approaches can be used and may have advantages in particular situations, the Bayesian approach is an intuitive and practical
Figure 1.2: Graphical illustration of how model-based methods are typically used to solve inverse problems in imaging applications. The measurements $Y$ of a physical system depend on an unknown image $X$. In addition, there are often unknown system-model parameters, $\phi$, and prior-model parameters, $\theta$. Model-based inversion works by searching for an input $\hat{x}$ that matches the measured output $y$, but also balances the need for the solution to be consistent with a prior model of the data.

Figure 1.3 graphically illustrates an intuitive argument for why model-based methods can dramatically improve the estimation of images from data. Obviously, an image formed by independent white noise has no chance of looking like a real natural image. But from a probabilistic perspective, there is a very, very small probability that a white noise image could appear to be the Mona Lisa, the famous painting of Leonardo da Vinci. Perhaps this is the visual parallel to the argument that a monkey randomly hitting the keys of a
1.1 What is Model-Based Image Processing?

Figure 1.3: Graphical illustration of how forward and prior models can interact synergistically to dramatically improve results. The green curve represents the thin manifold in which real images lie, and the gray line represents the thin manifold defined by noisy linear measurements. If the number of measurements, $M$, is less than the number of unknowns, $N$, then classical methods admit no unique solution. However, in this case model-based approaches can still provide a useful and unique solution at the intersection of the measurement and prior manifolds.

typewriter would eventually (almost surely in fact) produce Hamlet. While this may be true, we know that from a practical perspective any given white-noise image (or random sequence typed by a monkey) is much more likely to appear to be incoherent noise than a great work of art. This argument motivates the belief that real natural images have a great deal of structure that can be usefully exploited when solving inverse problems.

In fact, it can be argued that real natural images fall on thin manifolds in their embedded space. Such a thin image manifold is illustrated graphically in Figure 1.3 as a blurry green curve. So if $X \in [0, 1]^N$ is an image with $N$ pixels each of which fall in the interval $X_s \in [0, 1]$, then with very high probability, $X$ will fall within a very small fraction of the space in $[0, 1]^N$. Alternatively, a random sample from $[0, 1]^N$ will with virtual certainty never look like a natural image. It will simply look like noise.

In order to see that a natural images lives on a thin manifold, consider an image with one missing pixel, $X_s$. Then the conditional distribution of that
pixel given the rest of the image is \( p(x_s|x_r, r \neq s) \). In practice, a single missing pixel from a natural image can almost always be predicted with very high certainty. Say for example, that \( \mu_s = \mathbb{E}[x_s|x_r, r \neq s] \) is the conditional mean of \( X_s \) and \( \sigma_s = \text{Std}[x_s|x_r, r \neq s] \) is the conditional standard deviation. Then the “thickness” the manifold at the location \( X_r, r \neq s \) is approximately \( \sigma_s \).

Since \( X_s \) can be predicted with high certainty, we know that \( \sigma_s \ll 1 \), and the manifold is thin at that point. This in turn proves that a manifold containing real images, as illustrated in green, is thin in most places. However, while it is thin, it does in fact have some small thickness because we can almost never predict the missing pixel with absolute certainty. The situation is illustrated in Figure 1.3 for the case of a 2-D “image” formed by only two pixels \( X = [X_1, X_2] \).

Measurements typically constrain the solutions of an inverse problem to lie on a manifold that is, of course, data dependent. If the measurement is linear, then this thin measurement manifold is approximately a linear manifold (i.e., an affine subspace) except it retains some thickness due to the uncertainty introduced by measurement noise. So let \( Y = AX + W \), where \( Y \) is an \( M \) dimensional column vector of measurements, \( W \) is the measurement noise, and \( A \) is an \( M \times N \) matrix. Then each row of \( A \) represents a vector, illustrated by the red vectors of Figure 1.3 that projects \( X \) onto a measurement space. The resulting thin measurement manifold shown in gray is orthogonal to this projection, and its thickness is determined by the uncertainty due to noise in the measurement.

If the number of measurements, \( M \), is less than the dimension of the image, \( N \), then it would seem that the inverse problem can not be meaningfully solved because the solution can lie anywhere on the thin manifold defined by the measurement. In this situation, the prior information is absolutely essential because it effectively reduces the dimensionality of the unknown, \( X \). Figure 1.3 shows that even when the number of measurements is less than the number of unknowns, a good solution can be found at the intersection of the measurement and prior manifolds. This is why model-based methods can be so powerful in real problems. They allow one to solve inverse problems that otherwise would have no solution\(^2\).

\(^2\)Of course, other methods, most notably regularized inversion, have existed for a long time that allow for the solution of so-called ill-posed inverse problems. In fact, we include these approaches under the umbrella of model-based methods, but note that model-based methods typically emphasize the use of probabilistic models that are designed to accurately characterize the system and unknown image.
1.1 What is Model-Based Image Processing?

Nonetheless, there is a price to be paid for using model-based methods. First, model-based methods require the creation of accurate models of both the measurement system and the images, \( X \), being recovered. This is often a difficult and time consuming task. Second, the incorporation of a prior model necessarily introduces \textbf{bias} into the recovery of \( X \). Figure 1.4 conceptually represents the key tradeoff of variance versus bias that is at the heart of model-based approaches. When a researcher asks for a solution that makes no assumptions, they are implicitly asking for a solution with no bias in the estimate of \( \hat{X} \). However, estimators that achieve this goal often do it at the price of very large, or perhaps even infinite, variance in the final estimate. Roughly speaking, variance in the estimate can be viewed as noise, and bias can be viewed as systematic changes that would occur over many estimates whose noise was averaged away. So typical examples of bias are reconstruction error due to blur and other forms of excessive smoothing, or
systematic artifacts such as streaks or offsets.

Figure 1.4 graphically illustrates the tradeoff between variance and bias as a convex curve. As the bias is reduced, the variance must increase. The best solution depends on the specific application, but, for example, the minimum mean squared error solution occurs at the interaction of the red curve with a line of slope -1.

The example of Figure 1.4 illustrates a case where the variance can go to infinity as the bias goes to zero. In this case, a prior model is essential to achieving any useful solution. However, the prior model also introduces some systematic bias in the result; so the crucial idea is to use a prior model that minimizes any bias that is introduced. Traditional models that assume, for example, that images are band limited, are not the best models of real images in most applications, so they result in suboptimal results, such as the one illustrated with a star. These solutions are suboptimal because at a fixed bias, there exists a solution with less variance, and at a fixed variance, there exists a solution with lower bias.

Figure 1.5 shows the practical value of model-based methods [66]. This result compares state-of-the-art direct reconstruction results and model-based iterative reconstruction (MBIR) applied to data obtained from a General Electric 64 slice Volumetric Computed Tomography (VCT) scanner. The dramatic reduction in noise could not simply be obtained by post-processing of the image. The increased resolution and reduced noise is the result of the synergistic interaction between the measured data and the prior model.

1.2 How can I learn Model-Based Image Processing?

The remainder of this book presents the key mathematical and probabilistic techniques and theories that comprise model-based image processing. Perhaps the two most critical components are models of both systems and images, and estimators of $X$ based on those models. In particular, accurate image models are an essential component of model-based image processing due to the need to accurately model the thin manifold on which real images live.

---

3It can be proved that the optimal curve must be convex by randomizing the choice of any two algorithms along the curve.
1.2 How can I learn Model-Based Image Processing?

Figure 1.5: Comparison of state-of-the-art direct methods on the left to model-based iterative reconstruction (MBIR) on the right for the 3-D reconstruction of a medical image from data collected from a General Electric 64 slice Volumetric Computed Tomography (VCT) scanner. Notice that by using a accurate model of both the scanner and the image being reconstructed, it is possible to recover fine detail of the anatomy while simultaneously reducing noise.

Importantly, real images are not accurately modeled by Gaussian distributions. In fact, a histogram of the difference between adjacent pixel pairs in just about any natural image will clearly show a Laplacian distribution. Intuitively, the sharp edges in natural images caused by occlusion create statistics that are very non-Gaussian. Referring back to Figure 1.3, notice that the thin image manifold is represented as being curved. This nonlinear manifold is a direct result of the non-Gaussian nature of real images.

Another crucial requirement for image models is that they capture the non-causal nature of image structure. Most 1-D signals, such as audio or radio signals, can be modeled using causal structure. However, causal models break down in 2-D because they can no longer capture the naturally occurring characteristics of images. Unfortunately, we will see that in two or more dimensions non-casual models introduce very profound technical problems. This is because non-causal graphs in more than one dimension typically have loops, and these loops fundamentally break many of the algorithms and tools
of 1-D signal processing. Consequently, a major theme of this book will be to explain and understand the methods of Markov random fields, the central tool for the non-causal modeling of images and signals that live on 2-D lattices and loopy graphs.

<table>
<thead>
<tr>
<th>Gaussian</th>
<th>1-D Models</th>
<th>2-D Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Causal</td>
<td>AR (Sec. 3.3)</td>
<td>2-D AR (Sec. 3.4)</td>
</tr>
<tr>
<td>Non-Causal</td>
<td>1-D GMRF (Sec. 4.3)</td>
<td>2-D GMRF (Sec. 4.4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>non-Gaussian</th>
<th>1-D Models</th>
<th>2-D Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Causal</td>
<td>Markov Process</td>
<td>2-D Markov Process</td>
</tr>
<tr>
<td>Non-Causal</td>
<td>1-D MRF (Chap. 6)</td>
<td>2-D MRF (Chap. 6)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Discrete</th>
<th>1-D Models</th>
<th>2-D Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Causal</td>
<td>Markov Chain (Sec. 13.1)</td>
<td>2-D Markov Chain</td>
</tr>
<tr>
<td>Non-Causal</td>
<td>1-D MRF (Chap. 14)</td>
<td>2-D MRF (Chap. 14)</td>
</tr>
</tbody>
</table>

Table 1.1: A table of typical properties or characteristics that are often assumed in data models. The first, and perhaps simplest model we discuss is the 1-D Gaussian causal AR model of Section 3.3. From this first case, we generalize to include non-causal models in 2-D that take on both non-Gaussian and discrete values. The table lists out a total of 12 possible cases, of which we will cover 9 of these cases in some detail.

Table 1.1 lists out the various possible models, and shows the appropriate sections of the book where they are discussed. In fact, each topic represents a choice between three discrete categories: \{1-D, 2-D\}, \{causal, non-causal\}, and \{Gaussian, non-Gaussian, Discrete\}. Of course, real images are best modeled as 2-D, non-causal, and non-Gaussian; but we start with the simplest case of 1-D, causal, and Gaussian, and then work towards more complex models through the book. Towards the end of the book, we delve into discrete state models that are particularly useful for inverse problems involving the segmentation or classification of images.

Along the way, we introduce related theoretical and algorithmic techniques that are useful or even essential for solving model-based imaging applications. In particular, Chapters 5 and 7 present some of the basic methods of optimization in the context of model-based image processing. So the focus of this section is on understanding the advantages and disadvantages of different approaches, particularly when solving large problems.

Chapter 12 introduces the theory and practice of the EM algorithm, an essential tool in model-based image processing. The EM algorithm provides a general framework for estimating unknown model parameters when \(X\) is unknown. So for example, in Figures 1.1 and 1.2 the variables \(\phi\) and \(\theta\) denote unknown parameters of the system and image, respectively. In some cases,
these parameters can be determined in advanced, but in many cases they must be estimated from the data $Y$ as part of the inversion process. This is when the EM algorithm plays a critical role.

The final Chapters of 13 through 16 cover the basics of ergodic Markov chains and stochastic simulations. The techniques of stochastic simulation are particularly useful in applications such as image segmentation when $X$ is modeled as a discrete Markov random field. Stochastic simulation is also essential in the application of the EM algorithm when the so-called E-step can not be computed in closed form. This topic is covered in the section on stochastic EM.

Of course, all of this material depends on a solid understanding of the basic techniques of probability and estimation, so Chapter 2 presents a quick but comprehensive and hopefully truthful account of the essential concepts along with homework problems that reinforce the key ideas.
Chapter 2

Probability, Estimation, and Random Processes

This chapter provides a quick review of probability, estimation, and random processes. It also introduces many of the basic tools that will be needed for modeling physical data.

Two important themes throughout this book will be the use of both frequentist and Bayesian methods for the estimation of unknown quantities. In Section 2.3.1, the ubiquitous maximum likelihood (ML) estimator will be presented as an important example of the frequentist approach; and in Section 2.3.2, the minimum mean squared error (MMSE) estimator will be used to illustrate the Bayesian approach. Our goal is to show that each approach, frequentist or Bayesian, has its potential advantages, and the best choice tends to depend on the particular situation. Generally, we will see that the ML estimator tends to be a good choice when the ratio of the amount of data to the number of unknowns is much greater than 1, whereas the MMSE estimator tends to be a good choice when this ratio approaches or is less than 1.

2.1 Random Variables and Expectation

Let $X$ be a real valued random variable with cumulative distribution function (CDF) given by

$$F(t) \triangleq P\{X \leq t\}$$
where $P\{X \leq t\}$ is the probability of the event that $X$ is less than or equal to $t$. In fact, $F(t)$ is a valid CDF if and only if it has three properties: First, it must be a monotone increasing function of $t$; second, it must be right hand continuous; and third, must have limits of $\lim_{t\to\infty} F(t) = 1$ and $\lim_{t\to-\infty} F(t) = 0$.

If $F(t)$ is an absolutely continuous function, then it will have an associated probability density function (PDF), $p(t)$, such that

$$F(t) = \int_{-\infty}^{t} p(\tau) d\tau .$$

For most physical problems, it is reasonable to assume that such a density function exists, and when it does then it is given by

$$p(t) = \frac{dF(t)}{dt} .$$

Any function\(^1\) of a random variable is itself a random variable. So for example, if we define the new random variable $Y = g(X)$ where $g : \mathbb{R} \to \mathbb{R}$, then $Y$ will also be a random variable.

Armed with the random variable $X$ and its distribution, we may define the expectation as

$$\mathbb{E}[X] \triangleq \int_{-\infty}^{\infty} \tau dF(\tau) = \int_{-\infty}^{\infty} \tau p(\tau) d\tau .$$

The first integral form is known as a Lebesgue-Stieltjes integral and is defined even when the probability density does not exist. However, the second integral containing the density is perhaps more commonly used.

The expectation is a very basic and powerful tool which exists under very general conditions\(^2\). An important property of expectation, which directly results from its definition as an integral, is linearity.

**Property 2.1. Linearity of expectation** - For all random variables $X$ and $Y$,

$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y] .$$

\(^1\) Technically, this can only be a Lebesgue-measurable function; but in practice, measurability is a reasonable assumption in any physically meaningful situation.

\(^2\) In fact, for any positive random variable, $X$, the $E[X]$ takes on a well-defined value on the extended real line of $(-\infty, \infty]$. In addition, whenever $E[|X|] < \infty$, then $E[X]$ is real valued and well-defined. So we will generally assume that $E[|X|] < \infty$ for all random variables we consider.
Of course, it is also possible to specify the distribution of groups of random variables. So let $X_1, \ldots, X_n$ be $n$ random variables. Then we may specify the joint distribution of these $n$ random variables via the $n$-dimensional CDF given by

$$F(t_1, \cdots, t_n) = P\{X_1 \leq t_1, \cdots, X_n \leq t_n\}.$$ 

In this case, there is typically an associated $n$-dimensional PDF, $p(t_1, \cdots, t_n)$, so that

$$F(t_1, \cdots, t_n) = \int_{-\infty}^{t_1} \cdots \int_{-\infty}^{t_n} p(\tau_1, \cdots, \tau_n) d\tau_1 \cdots d\tau_n.$$ 

Again, any function of the vector $X = (X_1, \cdots, X_n)$ is then a new random variable. So for example, if $Y = g(X)$ where $g : \mathbb{R}^n \to \mathbb{R}$, then $Y$ is a random variable, and we may compute its expectation as

$$\mathbb{E}[Y] = \int_{\mathbb{R}^n} g(\tau_1, \cdots, \tau_n) p(\tau_1, \cdots, \tau_n) d\tau_1 \cdots d\tau_n.$$ 

If we have a finite set of random variables, $X_1, \cdots, X_n$, then we say the random variables are **jointly independent** if we can factor the CDF (or equivalently the PDF if it exists) into a product with the form,

$$F(t_1, \cdots, t_n) = F_{x_1}(t_1) \cdots F_{x_n}(t_n),$$

where $F_{x_k}(t_k)$ denotes the CDF for the random variable $X_k$. This leads to another important property of expectation.

**Property 2.2. Expectation of independent random variables** - If $X_1, \cdots, X_n$ are a set of jointly independent random variables, then we have that

$$\mathbb{E}\left[\prod_{k=1}^{n} X_k\right] = \prod_{k=1}^{n} \mathbb{E}[X_k].$$

So when random variables are jointly independent, then expectation and multiplication can be interchanged. Perhaps surprisingly, pair-wise independence of random variables does not imply joint independence. (See problem 3)

One of the most subtle and important concepts in probability is conditional probability and conditional expectation. Let $X$ and $Y$ be two random variables with a joint CDF given by $F(x, y)$ and **marginal CDF** given by
\( F_y(y) = \lim_{x \to \infty} F(x, y) \). The **conditional CDF** of \( X \) given \( Y = y \) is then any function, \( F(x|y) \), which solves the equation

\[
F(x, y) = \int_{-\infty}^{y} F(x|t)dF_y(t) .
\]

At least one solution to this equation is guaranteed to exist by an application of the famous Radon-Nikodym theorem. So the conditional CDF, \( F(x|y) \), is guaranteed to exist. However, this definition of the conditional CDF is somewhat unsatisfying because it does not specify how to construct \( F(x|y) \).

Fortunately, the conditional CDF can be calculated in most practical situations as

\[
F(x|y) = \frac{dF(x, y)}{dy} \left( \frac{dF(\infty, y)}{dy} \right)^{-1}.
\]

More typically, we will just work with probability density functions so that

\[
p(x, y) = \frac{d^2F(x, y)}{dxdy} \\
p_y(y) = \frac{dF(\infty, y)}{dy} .
\]

Then the PDF of \( X \) given \( Y = y \) is given by

\[
p(x|y) = \frac{p(x, y)}{p_y(y)} .
\]

We may now ask what is the conditional expectation of \( X \) given \( Y \)? It will turn out that the answer to this question has a subtle twist because \( Y \) is itself a random variable. Formally, the conditional expectation of \( X \) given \( Y \) is given by

\[
\mathbb{E}[X|Y] = \int_{-\infty}^{\infty} x dF(x|Y) ,
\]

or alternatively using the conditional PDF, it is given by

\[
\mathbb{E}[X|Y] = \int_{-\infty}^{\infty} xp(x|Y) dx .
\]

\footnote{Interestingly, the solution is generally not unique, so there can be many such functions \( F(x|y) \). However, it can be shown that if \( F(x|y) \) and \( F'(x|y) \) are two distinct solution, then the functions \( F(x|Y) \) and \( F'(x|Y) \) are almost surely equal.}

\footnote{In truth, this is an equality between two random variables that are almost surely equal.}
Notice that in both cases, the integral expression for $E[X|Y]$ is a function of the random variable $Y$. This means that the conditional expectation is itself a random variable!

The fact that the conditional expectation is a random variable is very useful. For example, consider the so-called **indicator function** denoted by

$$I_A(X) \triangleq \begin{cases} 
1 & \text{if } X \in A \\
0 & \text{otherwise}
\end{cases}.$$ 

Using the indicator function, we may express the conditional probability of an event, $\{X \in A\}$, given $Y$ as

$$P\{X \in A|Y\} = E[I_A(X)|Y] .$$  \hspace{1cm} (2.1)

Conditional expectation also has many useful properties. We list some below.

**Property 2.3. Filtration property of conditional expectation** - For all random variables $X$, $Y$, and $Z$


A special case of the filtration property is that $E[X] = E[E[X|Y]]$. This can be a useful relationship because sometimes it is much easier to evaluate an expectation in two steps, by first computing $E[X|Z]$ and then taking the expectation over $Z$. Another special case of filtration occurs when we use the relationship of equation (2.1) to express conditional expectations as conditional probabilities. In this case, we have that

$$E[ P \{X \in A|Y,Z\} | Y ] = P \{X \in A|Y\} .$$

Again, this relationship can be very useful when trying to compute conditional expectations.

**Property 2.4. Conditional expectation of known quantities** - For all random variables $X$, $Y$, and $Z$, and for all functions $f(\cdot)$

$$E[f(X)Z|X,Y] = f(X)E[Z|X,Y] ,$$

which implies that $E[X|X,Y] = X$. 

This property states the obvious fact that if we are given knowledge of $X$, then the value of $f(X)$ is known. For example, if we are told that the temperature outside is exactly 100 degrees Fahrenheit, then we know that the expectation of the outside temperature is 100 degrees Fahrenheit. Notice that since $E[X|Y] = f(Y)$ for some function $f(\cdot)$, Property 2.4 may be used to show that


Therefore, we know that for all random variables, $X$, $Y$, and $Z$


### 2.2 Some Commonly Used Distributions

Finally, we should introduce some common distributions and their associated notation. A widely used distribution for a random variable, $X$, is the **Gaussian distribution** denoted by

$$p(x) = \frac{1}{\sqrt{2\pi} \sigma^2} \exp \left\{ -\frac{1}{2\sigma^2} (x - \mu)^2 \right\}$$

where $\mu \in \mathbb{R}$ and $\sigma^2 > 0$ are known as the **mean** and **variance** parameters of the Gaussian distribution, and it can be easily shown that

$$E[X] = \mu \quad \quad E[(X - \mu)^2] = \sigma^2.$$

We use the notation $X \sim N(\mu, \sigma^2)$ to indicate that $X$ is a Gaussian random variable with mean $\mu$ and variance $\sigma^2$.

A generalization of the Gaussian distribution is the **multivariate Gaussian distribution**. We can use vector-matrix notation to compactly represent the distributions of multivariate Gaussian random vectors. Let $X \in \mathbb{R}^p$ denote a $p$-dimensional random column vector, and $X^t$ denote its transpose. Then the PDF of $X$ is given by

$$p(x) = \frac{1}{(2\pi)^{p/2}|R|^{-1/2}} \exp \left\{ -\frac{1}{2}(x - \mu)^t R^{-1}(x - \mu) \right\}, \tag{2.3}$$
2.2 Some Commonly Used Distributions

where $\mu \in \mathbb{R}^p$ is the mean vector and $R \in \mathbb{R}^{p \times p}$ is the covariance matrix of the multivariate Gaussian distribution. Without loss of generality, the covariance matrix can be assumed to be symmetric (i.e., $R = R^t$); however, in order to be a well defined distribution the covariance must also be positive definite, otherwise $p(x)$ cannot integrate to 1. More precisely, let $\mathbf{0}$ denote a column vector of 0’s. We say that a symmetric matrix is positive semi-definite if for all $y \in \mathbb{R}^p$ such that $y \neq \mathbf{0}$, then $||y||_R^2 \triangleq y^t R y \geq 0$. If the inequality is strict, then the matrix is said to be positive definite.

We will use the notation $X \sim N(\mu, R)$ to indicate that $X$ is a multivariate Gaussian random vector with mean $\mu$ and positive-definite covariance $R$, and in this case it can be shown that

$$
\mathbb{E}[X] = \mu
$$

$$
\mathbb{E}[(X - \mu)(X - \mu)^t] = R.
$$

A random vector with the distribution of equation (2.3) is also known as a jointly Gaussian random vector. It is important to note that a vector of Gaussian random variables is not always jointly Gaussian. So for example, it is possible for each individual random variable to have a marginal distribution that is Gaussian, but for the overall density not to have the form of (2.3). Furthermore, it can be shown that a random vector is jointly Gaussian, if and only if any linear transformation of the vector produces Gaussian random variables. For example, it is possible to construct two Gaussian random variables, $X_1$ and $X_2$, such that each individual random variable is Gaussian, but the sum, $Y = X_1 + X_2$, is not. In this case, $X_1$ and $X_2$ are not jointly Gaussian.

An important property of symmetric matrices is that they can be diagonalized using an orthonormal transformation. So for example, if $R$ is symmetric and positive definite, then

$$
R = E \Lambda E^t
$$

where $E$ is an orthonormal matrix whose columns are the eigenvectors of $R$ and $\Lambda$ is a diagonal matrix of strictly positive eigenvalues. Importantly, if $X$ is a zero-mean Gaussian random vector with covariance $R$, then we can

---

5 Furthermore, we say that a finite set of random vectors is jointly Gaussian if the concatenation of these vectors forms a single vector that is jointly Gaussian.
decorrelate the entries of the vector \( X \) with the transformation
\[
\tilde{X} = E^t X .
\]
In this case, \( \mathbb{E} [\tilde{X} \tilde{X}^t] = \Lambda \), so the components of the vector \( \tilde{X} \) are said to be **uncorrelated** since then \( \mathbb{E} [\tilde{X}_i \tilde{X}_j] = 0 \) for \( i \neq j \). Moreover, it can be easily proved that since \( \tilde{X} \) is jointly Gaussian with uncorrelated components, then its components must be jointly independent.

Jointly Gaussian random variables have many useful properties, one of which is listed below.

**Property 2.5.** Linearity of conditional expectation for Gaussian random variables - If \( X \in \mathbb{R}^n \) and \( Y \in \mathbb{R}^p \) are jointly Gaussian random vectors, then
\[
\mathbb{E}[X|Y] = AY + b
\]
\[
\mathbb{E}[(X - \mathbb{E}[X|Y])(X - \mathbb{E}[X|Y])^t|Y] = C ,
\]
where \( A \in \mathbb{R}^{n \times p} \) is a matrix, \( b \in \mathbb{R}^n \) is a vector, and \( C \in \mathbb{R}^{n \times n} \) is a positive-definite matrix. Furthermore, if \( X \) and \( Y \) are zero mean, then \( b = 0 \) and the conditional expectation is a linear function of \( Y \).

The **Bernoulli distribution** is an example of a discrete distribution which we will find very useful in image and signal modeling applications. If \( X \) has a Bernoulli distribution, then
\[
P\{X = 1\} = \theta
\]
\[
P\{X = 0\} = 1 - \theta ,
\]
where \( \theta \in [0,1] \) parameterizes the distribution. Since \( X \) takes on discrete values, its CDF has discontinuous jumps. The easiest solution to this problem is to represent the distribution with a **probability mass function (PMF)** rather than a probability density function. The PMF of a random variable or random vector specifies the actual probability of the random quantity taking on a specific value. If \( X \) has a Bernoulli distribution, then it is easy to show that its PMF may be expressed in the more compact form
\[
p(x) = (1 - \theta)^{1-x}\theta^x ,
\]
where \( x \in \{0,1\} \). Another equivalent and sometimes useful expression for the PMF of a Bernoulli random variable is given by
\[
p(x) = (1 - \theta)\delta(x - 0) + \theta\delta(x - 1) ,
\]
where the function $\delta(x)$ is 1 when its argument is 0, and 0 otherwise.

Regardless of the specific choice of distribution, it is often the case that a series of $n$ experiments are performed with each experiment resulting in an independent measurement from the same distribution. In this case, we might have a set of jointly independent random variables, $X_1, \cdots, X_n$, which are \textbf{independent and identically distributed (i.i.d.)}. So for example, if the random variables are i.i.d. Gaussian with distribution $N(\mu, \sigma^2)$, then their joint distribution is given by

$$p(x) = p(x_1, \cdots, x_n) = \left(\frac{1}{2\pi \sigma^2}\right)^{n/2} \exp\left\{-\frac{1}{2\sigma^2} \sum_{k=1}^{n} (x_k - \mu)^2\right\}.$$  

Notice that for notational simplicity, $x$ is used to denote the entire vector of measurements.

\textbf{Example 2.1.} Let $X_1, \cdots, X_n$ be $n$ i.i.d. Gaussian random vectors each with multivariate Gaussian distribution given by $N(\mu, R)$, where $\mu \in \mathbb{R}^p$ and $R \in \mathbb{R}^{p \times p}$. We would like to know the density function for the entire observation. In order to simplify notation, we will represent the data as a matrix

$$X = [X_1, \cdots, X_n].$$

Since the $n$ observations, $X_1, \cdots, X_n$, are i.i.d., we know we can represent the density of $X$ as the following product of densities for each vector $X_k$.

$$p(x) = \prod_{k=1}^{n} \frac{1}{(2\pi)^{p/2}} |R|^{-1/2} \exp\left\{-\frac{1}{2} (x_k - \mu)^t R^{-1} (x_k - \mu)\right\}$$

$$= \frac{1}{(2\pi)^{np/2}} |R|^{-n/2} \exp\left\{-\frac{1}{2} \sum_{k=1}^{n} (x_k - \mu)^t R^{-1} (x_k - \mu)\right\}.$$

This expression may be further simplified by defining the following two new sample statistics,

$$b = \sum_{k=1}^{n} x_k = x1,$$  \hspace{1cm} (2.4)

$$S = \sum_{k=1}^{n} x_k x_k^t = xx^t.$$.  \hspace{1cm} (2.5)
where \( \mathbf{1} \) is an \( n \) dimensional column vector of 1’s, and \( xx^t \) is the product of the matrix \( x \) with its transpose. In general, a statistic is any function of the observed data, but typically, statistics summarize the data in some useful way. Using these two statistics, the PDF of \( X \) may be written as

\[
p(x) = \frac{1}{(2\pi)^{np/2}|R|^{-n/2}} \exp \left\{ -\frac{1}{2} \text{tr} \left\{ SR^{-1} \right\} + b^t R^{-1} \mu - \frac{n}{2} \mu^t R^{-1} \mu \right\},
\]

where \( \text{tr}\{A\} \) denotes the trace of the matrix \( A \). In fact, this expression easily results by applying the very useful property that

\[
\text{tr}\{AB\} = \text{tr}\{BA\},
\]

for any two matrices \( A \in \mathbb{R}^{m \times n} \) and \( B \in \mathbb{R}^{n \times m} \). (See problem [12])

---

If we define the combined statistic \( T = (b, S) \) from equations (2.4) and (2.5), then \( T \) is said to be a sufficient statistic for the family of multivariate Gaussian distributions parameterized by \( \theta = (\mu, R) \). More precisely, a sufficient statistic, \( T \), is a statistic with the property that the associated distribution can be written in the form

\[
p(x|\theta) = h(x)g(T(x), \theta),
\]

for some functions \( h(\cdot) \) and \( g(\cdot, \cdot) \). (See Section 12.7.1 for more details.) Intuitively, a sufficient statistic for a distribution contains all the information that is necessary to estimate an unknown parameter \( \theta \).

---

**Example 2.2.** Let \( X_1, \ldots, X_n \) be \( n \) i.i.d. Bernoulli random variables, each with parameter \( \theta \). We would like to know the probability mass function for the entire observation. Again, since the \( n \) observations are i.i.d., we know that the PMF is given by

\[
p(x) = \prod_{k=1}^{n} \left\{ (1 - \theta)^{1-x_k} \theta^{x_k} \right\} = (1 - \theta)^{n-N_1} \theta^{N_1}
\]

where \( N_1 = \sum_{k=1}^{n} x_k \) is a sufficient statistic which counts the number of random variables \( X_k \) that are equal to 1.
2.3 Frequentist and Bayesian Estimators

Once we have decided on a model, our next step is often to estimate some information from the observed data. There are two general frameworks for estimating unknown information from data. We will refer to these two general frameworks as the Frequentist and Bayesian approaches. We will see that each approach is valuable, and the key is understanding what combination of approaches is best for a particular application.

The following subsections present each of the two approaches, and then provide some insight into the strengths and weaknesses of each approach.

2.3.1 Frequentist Estimation and the ML Estimator

In the frequentist approach, one treats the unknown quantity as a deterministic, but unknown parameter vector, $\theta \in \Omega$. So for example, after we observe the random vector $Y \in \mathbb{R}^p$, then our objective is to use $Y$ to estimate the unknown scalar or vector $\theta$. In order to formulate this problem, we will assume that the vector $Y$ has a PDF given by $p_\theta(y)$ where $\theta$ parameterizes a family of density functions for $Y$. We may then use this family of distributions to determine a function, $T : \mathbb{R}^n \to \Omega$, that can be used to compute an estimate of the unknown parameter as

\[ \hat{\theta} = T(Y) . \]

Notice, that since $T(Y)$ is a function of the random vector $Y$, the estimate, $\hat{\theta}$, is a random vector. The mean of the estimator, $\bar{\theta}$, can be computed as

\[ \bar{\theta} = \mathbb{E}_\theta [\hat{\theta}] = \mathbb{E}_\theta [T(Y)] = \int_{\mathbb{R}^n} T(y)p_\theta(y)dy . \]

The difference between the mean of the estimator and the value of the parameter is known as the bias and is given by

\[ \text{bias}_\theta = \bar{\theta} - \theta . \]

Similarly, the variance of the estimator is given by

\[ \text{var}_\theta = \mathbb{E}_\theta \left[ \left( \hat{\theta} - \bar{\theta} \right)^2 \right] , \]
and it is easily shown that the mean squared error (MSE) of the estimate is then given by

$$\text{MSE}_\theta = E_{\theta} \left[ (\hat{\theta} - \theta)^2 \right] = \text{var}_\theta + (\text{bias}_\theta)^2.$$  

Since the bias, variance, and MSE of the estimator will depend on the specific value of $\theta$, it is often unclear precisely how to compare the accuracy of different estimators. Even estimators that seem quite poor may produce small or zero error for certain values of $\theta$. For example, consider the estimator which is fixed to the value $\hat{\theta} = 1$, independent of the data. This would seem to be a very poor estimator, but it has an MSE of 0 when $\theta = 1$.

Many important properties of an estimator depend on its behavior as the amount of observed data tends toward infinity. So, imagine that we have a series of $n$ independent observations denoted by $Y = (Y_1, \cdots, Y_n)$, and let $\hat{\theta}_n(Y)$ be an estimate of the parameter $\theta$ that uses all $n$ samples. We say that the estimate $\hat{\theta}_n$ is consistent if it converges to the true value in probability as $n$ goes to infinity. If an estimator is not consistent, this means that even with arbitrarily large quantities of data, the estimate is not guaranteed to approach the true value of the parameter. Consistency would seem to be the least we would expect of an estimator, but we will later see that even some very intuitive estimators are not always consistent.

Ideally, it would be best if one could select an estimator which has uniformly low bias and/or variance for all values of $\theta$. This is not always possible, but when it is we have names for such estimators. For example, $\hat{\theta}$ is said to be an unbiased estimator if for all values of $\theta$ the bias is zero, i.e. $\theta = \bar{\theta}$. If in addition, for all values of $\theta$, the variance of an estimator is less than or equal to that of all other unbiased estimators, then we say that the estimator is a uniformly minimum variance unbiased (UMVU) estimator.

There are many excellent estimators that have been proposed through the years for many different types of problems. However, one very widely used Frequentist estimator is known as the maximum likelihood (ML)
estimator given by
\[
\hat{\theta} = \arg \max_{\theta \in \Omega} p_{\theta}(Y) \\
= \arg \max_{\theta \in \Omega} \log p_{\theta}(Y) .
\]
where the notation “arg max” denotes the value of the argument that achieves the global maximum of the function. Notice that these formulas for the ML estimate actually use the random variable \( Y \) as an argument to the density function \( p_{\theta}(y) \). This implies that \( \hat{\theta} \) is a function of \( Y \), which in turn means that \( \hat{\theta} \) is a random variable.

When the density function, \( p_{\theta}(y) \), is a continuously differentiable function of \( \theta \), then a necessary condition when computing the ML estimate is that the gradient of the likelihood is zero.
\[
\nabla_{\theta} p_{\theta}(Y)|_{\theta=\hat{\theta}} = 0 .
\]

While the ML estimate is generally not unbiased, it does have a number of desirable properties. It can be shown that under relatively mild technical conditions, the ML estimate is both consistent and asymptotically efficient. This means that the ML estimate attains the Cramér-Rao bound asymptotically as the number of independent data samples, \( n \), goes to infinity [61]. Since this Cramér-Rao bound is a lower bound on the variance of an unbiased estimator, this means that the ML estimate is about as good as any unbiased estimator can be. So if one has no idea what values of \( \theta \) may occur, and needs to guarantee good performance for all cases, then the ML estimator is usually a good choice.

**Example 2.3.** Let \( \{Y_k\}_{k=1}^n \) be i.i.d. random variables with Gaussian distribution \( N(\theta, \sigma^2) \) and unknown mean parameter \( \theta \). For this case, the logarithm of the density function is given by
\[
\log p_{\theta}(Y) = -\frac{1}{2\sigma^2} \sum_{k=1}^n (Y_k - \theta)^2 - \frac{n}{2} \log (2\pi\sigma^2) .
\]
Differentiating the log likelihood results in the following expression.
\[
\left. \frac{d \log p_{\theta}(Y)}{d\theta} \right|_{\theta=\hat{\theta}} = \frac{1}{\sigma^2} \sum_{k=1}^n (Y_k - \hat{\theta}) = 0
\]
From this we obtain the ML estimate for $\theta$.

$$\hat{\theta} = \frac{1}{n} \sum_{k=1}^{n} Y_k$$

Notice that, by the following argument, this particular ML estimate is unbiased.

$$\mathbb{E}_\theta[\hat{\theta}] = \mathbb{E}_\theta\left[\frac{1}{n} \sum_{k=1}^{n} Y_k\right] = \frac{1}{n} \sum_{k=1}^{n} \mathbb{E}_\theta[Y_k] = \frac{1}{n} \sum_{k=1}^{n} \theta = \theta$$

Moreover, for this special case, it can be shown that the ML estimator is also the UMVU estimator \cite{61}.

---

**Example 2.4.** Again, let $X_1, \cdots, X_n$ be $n$ i.i.d. Gaussian random vectors each with multivariate Gaussian distribution given by $N(\mu, R)$, where $\mu \in \mathbb{R}^p$. We would like to compute the ML estimate of the parameter vector $\theta = [\mu, R]$. Using the statistics $b$ and $S$ from Example 2.1, we can define the sample mean and covariance given by

$$\hat{\mu} = b/n$$

$$\hat{R} = (S/n) - \hat{\mu}\hat{\mu}^t,$$

then by manipulation of equation (2.6) the probability distribution of $X$ can be written in the form

$$p(x|\theta) = \frac{1}{(2\pi)^{np/2}|R|^{-n/2}} \exp\left\{ -\frac{n}{2} \text{tr}\left\{ \hat{R}R^{-1}\right\} - \frac{n}{2}(\hat{\mu} - \mu)^t R^{-1}(\hat{\mu} - \mu) \right\}$$

Maximizing this expression with respect to $\theta$ then results in the ML estimate of the mean and covariance given by

$$\arg \max_{[\mu, R]} p(x|\mu, R) = [\hat{\mu}, \hat{R}] .$$

So from this we see that the ML estimate of the mean and covariance of a multivariate Gaussian distribution is simply given by the sample mean and covariance of the observations.
2.3 Frequentist and Bayesian Estimators

Example 2.5. Let $X_1, \ldots, X_n$ be $n$ i.i.d. Bernoulli random variables each with parameter $\theta$. From Example 2.2, we know that

$$\log p(x|\theta) = (n - N_1) \log(1 - \theta) + N_1 \log \theta,$$

where $N_1$ is a sufficient statistic for the parameter $\theta$. We can then compute the ML estimate by differentiating with respect to $\theta$ and setting the expression to 0.

$$\frac{\partial \log p(x|\theta)}{\partial \theta} = -\frac{n - N_1}{1 - \theta} + \frac{N_1}{\theta} = 0$$

This yields the ML estimate for the parameter of the Bernoulli distribution.

$$\hat{\theta} = \frac{N_1}{n}$$

This very intuitive results simply says that the ML estimate is given by the observed fraction of times that $X_n$ is 1.

2.3.2 Bayesian Estimation and the MAP Estimator

Since the ML estimator is asymptotically efficient, it is guaranteed to do well for all values of the unknown parameter $\theta$, as the amount of data grows towards infinity. So it might seem that nothing more can be done.

However, Bayesian methods attempt to exceed the accuracy of Frequentist estimators, such as the ML estimate, by making assumptions about values of the parameters that are most likely to occur. In practice, we will see that Bayesian estimation methods are most useful when the amount of data is relatively small compared to the dimension of the unknown parameter.

In the Bayesian approach, we model the unknown quantity as random, rather than deterministic. In order to emphasize this distinction, we will use the random variable $X$ to denote the unknown quantity in Bayesian estimation, as opposed to the unknown parameter, $\theta$, used in the frequentist approach. As before, the estimator is then a function of the observations with the form

$$\hat{X} = T(Y),$$

and once again the estimate, $\hat{X}$, is a random vector.
The good news with Bayesian estimators is that once we make the assumption that $X$ is random, then the design of estimators is conceptually straightforward, and the MSE of our estimator can be reduced by estimating values of $X$ that are more probable to occur. However, the bad news is that when we assume that $X$ is a random vector, we must select some distribution for it. In some cases, the distribution for $X$ may be easy to select, but in many cases, it may be very difficult to select a reasonable distribution or model for $X$. And in some cases, a poor choice of this prior distribution can severely degrade the accuracy of the estimator.

In any case, let us assume that the joint distribution of both $Y$ and $X$ is known, and let $C(x, \hat{x}) \geq 0$ be the cost of choosing $\hat{x}$ as our estimate if $x$ is the correct value of the unknown. If our estimate is perfect, then $\hat{x} = x$, which implies that $C(x, \hat{x}) = C(x, x) = 0$; so the cost is zero. Of course, in most cases our estimates will not be perfect, so our objective is to select an estimator that minimizes the expected cost. For a given value of $X = x$, the expected cost is given by

$$\bar{C}(x) = \mathbb{E}[C(x, T(Y))|X = x]$$

$$= \int_{\mathbb{R}^n} C(x, T(y)) p_{y|x}(y|x) dy$$

where $p_{y|x}(y|x)$ is the conditional distribution of the data, $Y$, given the unknown, $X$. Here again, the cost is a function of the unknown, $X$, but we can remove this dependency by taking the expectation to form what is known as the **Bayes’ risk**.

$$\text{Risk} = \mathbb{E}[C(X, T(Y))]$$

$$= \mathbb{E}[\mathbb{E}[C(X, T(Y))|X]]$$

$$= \mathbb{E}[\bar{C}(X)]$$

$$= \int \bar{C}(x) p_x(x) dx$$

Notice, that the risk is not a function of $X$. This stands in contrast to the Frequentist case, where the MSE of the estimator was a function of $\theta$. However, the price we pay for removing this dependency is that the risk now depends critically on the choice of the density function $p_x(x)$, which specifies the distribution on the unknown $X$. The distribution of $X$ is known as the
prior distribution since it is the distribution that $X$ is assumed to have prior to the observation of any data.

Of course, the choice of a specific cost function will determine the form of the resulting best estimator. For example, it can be shown that when the cost is squared error, so that $C(x, \hat{x}) = |x - \hat{x}|^2$, then the optimal estimator is given by the conditional expectation

$$\hat{X}_{MMSE} = \mathbb{E}[X|Y],$$

and when the cost is absolute error, so that $C(x, \hat{x}) = |x - \hat{x}|$, then the optimal estimator is given by the conditional median.

$$\int_{-\infty}^{\hat{x}_{\text{median}}} p_{x|y}(x|Y)dx = \frac{1}{2}$$

Another important case, that will be used extensively in this book, is when the cost is given by

$$C(x, \hat{x}) = 1 - \delta(x - \hat{x}) = \begin{cases} 0 & \text{if } x = \hat{x} \\ 1 & \text{if } x \neq \hat{x} \end{cases}.$$  

Here, the cost is only zero when $x = \hat{x}$, and otherwise the cost is fixed at 1. This is a very pessimistic cost function since it assigns the same value whether the error is very small or very large. The optimal estimator for this cost function is given by the so-called maximum a posteriori (MAP) estimate,

$$\hat{X}_{MAP} = \arg \max_{x \in \Omega} p_{x|y}(x|Y),$$  

(2.11)

where $\Omega$ is the set of feasible values for $X$, and the conditional distribution $p_{x|y}(x|y)$ is known as the posterior distribution because it specifies the distribution of the unknown $X$ given the observation $Y$.

Typically, it is most natural to express the posterior probability in terms of the forward model, $p_{y|x}(y|x)$, and the prior model, $p_x(x)$, using Bayes’ rule.

$$p_{x|y}(x|y) = \frac{p_{y|x}(y|x)p_x(x)}{p_y(y)},$$  

(2.12)

In practice, the forward model can typically be formulated from a physical model of how the measurements, $Y$, are obtained given an assumed image, $X$. 
The prior model, $p_x(x)$, is then selected to accurately capture the variations that occur in the images being estimated.

With this framework, the expression for the MAP estimate can then be reformulated as

$$
\hat{X}_{MAP} = \arg \max_{x \in \Omega} \log p_{x|y}(x|Y)
$$

where

$$
\begin{align*}
\hat{X}_{MAP} &= \arg \max_{x \in \Omega} \{ \log p_{y|x}(y|x) + \log p_x(x) - \log p_y(y) \} \\
&= \arg \min_{x \in \Omega} \{ -\log p_{y|x}(y|x) - \log p_x(x) \}
\end{align*}
$$

In this form, it is clear that the MAP estimate balances the need to fit the data (i.e., minimize the forward model’s negative log likelihood) with fitting our prior model of how the reconstruction should appear (i.e., minimize the negative log prior term). This balance of fitting the observed data while remaining consistent with our the prior model represents the core of what the Bayesian framework attempts to achieve.

**Example 2.6.** Let $X$ and $W$ be Gaussian random variables with distributions $N(0, \sigma_x^2)$ and $N(0, \sigma_w^2)$, respectively. Furthermore, let $Y = X + W$. Our task is then to estimate $X$ from the observations of $Y$.

Using the definition of conditional probability, we may express the joint distribution of $X$ and $Y$ as

$$
p_{y,x}(y, x) = p_{x|y}(x|y)p_y(y) = p_{y|x}(y|x)p_x(x)
$$

Reorganizing terms results in Bayes’ rule, which is the source of the terminology **Bayesian estimation**.

$$
p_{x|y}(x|y) = \frac{p_{y|x}(y|x)p_x(x)}{p_y(y)}
$$

One approach to computing the posterior distribution is to directly evaluate Bayes’ rule. However, this is somewhat cumbersome in this case, and for some problems, it will actually be impossible. An alternative approach is to consider the posterior as

$$
p_{x|y}(x|y) = \frac{1}{z_y} p_{y,x}(y, x)
$$
where \( z_y \) is a normalizing constant or **partition function** for our problem. Since the posterior distribution is a probability density in \( x \), the right side of equation (2.16) must integrate to 1 as a function of \( x \). This means that \( z_y \) is uniquely specified as the normalizing constant of the distribution. Notice, that the partition function may depend on \( y \), but it may not depend on \( x \).

Using this approach, we first evaluate the conditional data distribution, \( p_{y|x}(y|x) \), and prior distribution, \( p_x(x) \), for our problem. The prior is assumed to be

\[
p_x(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp \left\{ -\frac{1}{2\sigma_x^2}x^2 \right\},
\]

and the conditional density of the data is given by

\[
p_{y|x}(y|x) = \frac{1}{\sqrt{2\pi\sigma_w^2}} \exp \left\{ -\frac{1}{2\sigma_w^2}(y - x)^2 \right\}.
\]

This means the joint distribution of \( X \) and \( Y \) can be written in the form

\[
p_{x|y}(x|y) = \frac{1}{z_y} \exp \left\{ -\frac{1}{2\sigma_w^2}(y - x)^2 - \frac{1}{2\sigma_x^2}x^2 \right\}
= \frac{1}{z'_y} \exp \left\{ a(x - b)^2 \right\}, \tag{2.17}
\]

where \( a \) and \( b \) are two appropriately chosen parameters. As with the partition function \( z_y \), the parameters \( a \) and \( b \) can be functions of \( y \), but they can not be functions of \( x \). Also notice that in general \( z_y, z'_y, \) and \( z''_y \) are not equal, but in each case, the partition function collects together all the multiplicative factors in the expression that do not depend on \( x \).

We can solve for the values of the parameters \( a \) and \( b \) by taking the first two derivatives of the logarithms of the expressions and setting the derivatives to be equal. For the first derivative, this results in the following calculation.

\[
\frac{d}{dx} \log \left( \frac{1}{z'_y} \exp \left\{ -\frac{1}{2\sigma_w^2}(y - x)^2 - \frac{1}{2\sigma_x^2}x^2 \right\} \right)
= \frac{d}{dx} \log \left( \frac{1}{z''_y} \exp \left\{ a(x - b)^2 \right\} \right)
\]

\[
\frac{d}{dx} \left( -\frac{1}{2\sigma_w^2}(y - x)^2 - \frac{1}{2\sigma_x^2}x^2 - \log z'_y \right)
= \frac{d}{dx} \left( a(x - b)^2 - \log z''_y \right)
\]

\[
-\frac{1}{\sigma_w^2}(x - y) - \frac{1}{\sigma_x^2}x = 2a(x - b)
\]
Differentiating this expression a second time yields the result that

\[ a = -\frac{1}{2} \left( \frac{1}{\sigma_w^2} + \frac{1}{\sigma_x^2} \right), \]

and solving for \( b \) yields

\[ b = \left( \frac{\sigma_x^2}{\sigma_x^2 + \sigma_w^2} \right) y. \]

In fact, looking at the expression of equation (2.17), one can see that the parameters \( b \) and \( a \) specify the posterior mean and variance, respectively. More specifically, if we define \( \mu_{x|y} \) to be the mean of the posterior density and \( \sigma_{x|y}^2 \) to be its variance, then we have the relationships that

\[
-\frac{1}{2\sigma_{x|y}^2} = a = -\frac{1}{2} \left( \frac{1}{\sigma_w^2} + \frac{1}{\sigma_x^2} \right) = -\frac{1}{2} \left( \frac{\sigma_w^2 + \sigma_x^2}{\sigma_w^2 \sigma_x^2} \right)
\]

\[
\mu_{x|y} = b = \left( \frac{\sigma_x^2}{\sigma_x^2 + \sigma_w^2} \right) y.
\]

Solving for the conditional mean and variance, we can then write an explicit expression for the conditional distribution of \( x \) given \( y \) as

\[
p_{x|y}(x|y) = \frac{1}{\sqrt{2\pi\sigma_{x|y}^2}} \exp \left\{ -\frac{1}{2\sigma_{x|y}^2} (x - \mu_{x|y})^2 \right\}, \quad (2.18)
\]

where

\[
\mu_{x|y} = \frac{\sigma_x^2}{\sigma_x^2 + \sigma_w^2} y
\]

\[
\sigma_{x|y}^2 = \frac{\sigma_x^2 \sigma_w^2}{\sigma_x^2 + \sigma_w^2}.
\]

While simple, the result of Example 2.6 is extremely valuable in providing intuition about the behavior of Bayesian estimates. First, from the form of (2.18), we can easily see that the MMSE estimate of \( X \) given \( Y \) is given by

\[
\hat{X}_{MMSE} = \mathbb{E}[X|Y] = \frac{\sigma_x^2}{\sigma_x^2 + \sigma_w^2} Y = \frac{1}{1 + \frac{\sigma_w^2}{\sigma_x^2}} Y.
\]
Notice that the MMSE estimate is given by \( Y \) multiplied by a constant, \( 0 \leq \frac{\sigma_x^2}{\sigma_x^2 + \sigma_w^2} < 1 \). In other words, the Bayesian estimate is simply an attenuated version of \( Y \). Moreover, the degree of attenuation is related to the ratio \( \frac{\sigma_w^2}{\sigma_x^2} \), which is the **noise-to-signal ratio**. When the noise-to-signal ratio is low, the attenuation is low because \( Y \) is a good estimate of \( X \); but when the noise-to-signal ratio is high, then the attenuation is great because \( Y \) is only loosely correlated with \( X \). While this attenuation can reduce mean-square error, it also results in an estimate which is biased because

\[
\mathbb{E} \left[ \hat{X}_{MMSE} | X \right] = \frac{\sigma_x^2}{\sigma_x^2 + \sigma_w^2} X < X .
\]

This scalar result can also be generalized to the case of jointly Gaussian random vectors to produce a result of general practical value as shown in the following example.

**Example 2.7.** Let \( X \sim N(0, R_x) \) and \( W \sim N(0, R_w) \) be independent Gaussian random vectors, and let \( Y = X + W \). Then the conditional distribution of \( X \) given \( Y \) can be calculated in much the same way as it was done for Example 2.6. The only difference is that the quadratic expressions are generalized to matrix vector products and quadratic forms. (See problem 13) Under these more general conditions, the conditional distribution is given by

\[
p_{x|y}(x|y) = \frac{1}{(2\pi)^{n/2} |R_x|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu_{x|y})^t R_{x|y}^{-1} (x - \mu_{x|y}) \right\} ,
\]

where

\[
\mu_{x|y} = \left( R_x^{-1} + R_w^{-1} \right)^{-1} R_w^{-1} y
\]

\[
= \left( I + R_w R_x^{-1} \right)^{-1} y
\]

\[
= R_x (R_x + R_w)^{-1} y
\]

\[
R_{x|y} = \left( R_x^{-1} + R_w^{-1} \right)^{-1}
\]

\[
= R_x (R_x + R_w)^{-1} R_w ,
\]

and the MMSE estimate of \( X \) given \( Y \) is then given by

\[
\hat{X}_{MMSE} = \mathbb{E}[X|Y] = R_x (R_x + R_w)^{-1} Y .
\]
At this point, there are two interesting observations to make, which are both particular to the case of jointly Gaussian random variables. First, for jointly Gaussian random quantities, the posterior mean is a linear function of the data $y$, and the posterior variance is not a function of $y$. These two properties will not hold for more general distributions. Second, when the observation, $Y$, and unknown, $X$, are jointly Gaussian, then the posterior distribution is Gaussian, so it is both a symmetric and unimodal function of $x$. Therefore, for jointly Gaussian random quantities, the conditional mean, conditional median, and MAP estimates are all identical.

$$\hat{X}_{\text{median}} = \hat{X}_{\text{MMSE}} = \hat{X}_{\text{MAP}}$$

However, for more general distributions, these estimators will differ.

### 2.3.3 Comparison of the ML and MMSE Estimators

In order to better understand the advantages and disadvantages of Bayesian estimators, we explore a simple example. Consider the case when $X$ is a Gaussian random vector with distribution $N(0, R_x)$ and $Y$ is formed by

$$Y = X + W$$

where $W$ is a Gaussian random vector with distribution $N(0, \sigma_w^2 I)$. So in this case, $Y$ is formed by adding independent Gaussian noise with variance $\sigma_w^2$ to the components of the unknown vector $X$. Furthermore, let us assume that the covariance of $X$ is formed by

$$[R_x]_{i,j} = \frac{1}{1 + |\frac{i-j}{10}|^2}, \quad (2.21)$$

such that $\sigma_w = 0.75$ and the dimension is $p = 50$. Using this model, we can generate a pseudo-random vector on a computer from the distribution of $X$. (See problem [7]) Such a vector is shown in the upper left plot of Figure 2.1. Notice, that $X$ is a relatively smooth function. This is a result of our particular choice of the covariance $R_x$. More specifically, because the covariance drops off slowly as the $|i - j|$ increases, nearby samples will be highly correlated, and it is probable (but not certain) that the function will
be smooth. However, once we add noise to the samples of $X$, correlation between nearby samples is largely lost. So the signal, $Y$, shown in Figure 2.1 is quite noisy compared to $X$.

For this problem, the ML estimate of $X$ given $Y$ is simply $Y$ itself! To see this, we simply apply the definition of the ML estimate keeping in mind that, for this approach, $X$ is considered to be a deterministic parameter rather than a random vector.

$$
\hat{X}_{ML} = \arg \max_x \left\{ \frac{1}{(2\pi \sigma_w^2)^{p/2}} \exp \left\{ -\frac{1}{2\sigma_w^2} (Y - x)^t(Y - x) \right\} \right\} = Y
$$

Alternatively, the MMSE estimate discussed in Example 2.7 can be computed using the expression of equation (2.20).

Figure 2.1 shows the plots of the original signal, noisy signal, and both the ML and MMSE estimates. For this example, the MMSE estimate is much better than the ML estimate because we have considerable knowledge about the behavior of $X$. While the ML estimate is unbiased, it does not use this prior knowledge, so it produces an unsatisfying result. Moreover, as we mentioned in the introduction, the ML estimator tends to be most suitable when the number of parameters to be estimated is much less than the number of data points. However, in this example there is only one data point for each unknown parameter; so this situation is not favorable to the ML estimate.

Alternatively, the MMSE estimate is not perfect, but it does seem to capture the essential trends in the true signal $X$. However, the price we have paid for this better estimate is that we are making strong assumptions about the nature of $X$, and if those assumptions are not true then the estimate may be very poor. In addition, the MMSE estimate will be biased. That is to say that the shape of the signal $\hat{X}_{MMSE}$ will be smoother than the actual signal. The tradeoff between bias and variance of an estimator is a fundamental one, which typically cannot be avoided.

### 2.4 Discrete-Time Random Processes

Now that we have introduced a framework for random variables, we can use sequences of random variables to build **discrete-time random processes**.
So for each integer $n \in \mathbb{Z}$, let $X_n$ be a random variable. Then we say that $X_n$ is a discrete-time random process.

In order to fully specify the distribution of a random process, it is necessary to specify the joint distribution of all finite subsets of the random variables. To do this, we can define the following finite-dimensional distribution

$$F_{m,k}(t_m, \cdots, t_k) = P \{ X_m \leq t_m, X_{m+1} \leq t_{m+1}, \cdots, X_k \leq t_k \},$$

where $m$ and $k$ are any integers with $m \leq k$. The extension theorem (see for example [70] page 3) guarantees that if the set of finite-dimensional distributions is consistent for all $m, k \in \mathbb{Z}$, then the distribution of the infinite dimensional random process is well defined. So the set of all finite dimensional distributions specify the complete distribution of the random process. Furthermore, if for all $m \leq k$, the vector $[X_m, X_{m+1}, \cdots, X_k]^T$ is jointly Gaussian, then we say that the $X_m$ is a Gaussian random process.
Often it is reasonable to assume that the properties of random processes do not change over time. Of course, the actual values of the random process change, but the idea is that the statistical properties of the random process are invariant. This type of time-invariance of random processes is known as **stationarity**. As it turns out, there are different ways to specify stationarity for a random process. Perhaps the most direct approach is to specify that the distribution of the random process does not change when it is shifted in time. This is called **strict-sense stationary**, and it is defined below.

**Definition 1. Strict-Sense Stationary Random Process**
A 1-D discrete-time random process $X_n$ is said to be strict-sense stationary if the finite dimensional distributions of the random process $X_n$ are equal to the finite dimensional distributions of the random process $Y_n = X_{n-k}$ for all integers $k$. This means that

$$F_{m,m+k}(t_0, t_1, \cdots, t_k) = F_{0,k}(t_0, t_1, \cdots, t_k),$$

for all integers $m$ and $k \geq 1$ and for all values of $t_0, t_1, \cdots, t_k \in \mathbb{R}$.

Strict-sense stationarity is quite strong because it depends on verifying that all finite-dimensional distributions are equal for both the original random process and its shifted version. In practice, it is much easier to check the second order properties of a random processes, than it is to check all its finite-dimensional distributions. To this end, we first define the concept of a **second-order random process**.

**Definition 2. Second Order Random Process**
A 1-D discrete-time random process $X_n$ is said to be a second order random process if for all $n,k \in \mathbb{Z}$, $\mathbb{E}[|X_nX_k|] < \infty$, i.e. all second order moments are finite.

If a random process is second order, then it has a well defined mean, $\mu_n$, and **time autocovariance**, $R(n,k)$, given by the following expressions.

$$\mu_n = \mathbb{E}[X_n]$$
$$R(n,k) = \mathbb{E}[(X_n - \mu_n)(X_k - \mu_k)]$$

Armed with these concepts, we can now give the commonly used definition of **wide-sense stationarity**.
Definition 3. Wide-Sense Stationary Random Process

A 1-D second-order discrete-time random process \( X_n \) is said to be wide-sense stationary if for all \( n, k \in \mathbb{Z} \), the \( \mathbb{E}[X_n] = \mu \) and \( \mathbb{E}[(X_n - \mu)(X_k - \mu)] = R(n - k) \) where \( \mu \) is a scalar constant and \( R(n) \) is a scalar function of \( n \).

So a discrete-time random process is wide-sense stationary if its mean is constant and its time autocovariance is only a function of \( n - k \), the difference between the two times in question.

Reversibility is another important invariance property which holds when a strict-sense stationary random process, \( X_n \), and its time-reverse, \( X_{-n} \), have the same distribution. The formal definition of a reversible random process is listed below.

Definition 4. Reversible Random Process

A 1-D discrete-time random process \( X_n \) is said to be reversible if its finite dimensional distributions have the property that

\[
F_{0,k}(t_0, t_1, \cdots, t_k) = F_{m,m+k}(t_k, t_{k-1}, \cdots, t_0),
\]

for all integers \( m \) and \( k \geq 1 \) and for all values of \( t_0, t_1, \cdots, t_k \in \mathbb{R} \).

Gaussian random processes have some properties which make them easier to analyze than more general random processes. In particular, the mean and time autocovariance for a Gaussian random process provides all the information required to specify its finite-dimensional distributions. In practice, this means that the distribution of any Gaussian random process is fully specified by its mean and autocovariance, as stated in the following property.

Property 2.6. Autocovariance specification of Gaussian random processes - The distribution of a discrete-time Gaussian random process, \( X_n \), is uniquely specified by its mean, \( \mu_n \), and time autocovariance, \( R(n, k) \).

If a Gaussian random process is also stationary, then we can equivalently specify the distribution of the random process from its power spectral density. In fact, it turns out that the power spectral density of a zero-mean wide-sense stationary Gaussian random process is given by the Discrete-time Fourier transform (DTFT) of its time autocovariance. More formally, if we define \( S_X(e^{j\omega}) \) to be the expected power spectrum of the signal,
$X_n$, then the **Wiener-Khinchin theorem** states that for a zero-mean wide-

sense stationary random process the power spectrum is given by the following

expression,

\[ S_X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} R(n)e^{-j\omega n} \]

where the right-hand side of the equation is the DTFT of the time autocovariance $R(n)$.

Since the DTFT is an invertible transform, knowing $S_X(e^{j\omega})$ is equivalent
to knowing the time autocovariance $R(n)$. Therefore the power spectrum also
fully specifies the distribution of a zero-mean wide-sense stationary Gaussian
random process. The next property summarizes this important result.

**Property 2.7. Power spectral density specification of a zero-mean wide-sense
stationary Gaussian random process** - The distribution of a wide-sense sta-
tionary discrete-time Gaussian random process, $X_n$, is uniquely specified by
its power spectral density, $S_X(e^{j\omega})$.

Finally, we consider reversibility of Gaussian random processes. So if $X_n$
is a wide-sense stationary Gaussian random process, we can define its time-reversed version as $Y_n = X_{-n}$. Then by the definition of time autocovariance, we know that

\[
R_Y(n) = E[(Y_k - \mu)(Y_{k+n} - \mu)] = E[(X_{-k} - \mu)(X_{-k-n} - \mu)] = E[(X_{l+n} - \mu)(X_l - \mu)] = R_X(n)
\]

where $l = -k - n$, and $\mu$ is the mean of both $X_n$ and $Y_n$. From this, we
know that $X_n$ and its time reverse, $Y_n$, have the same autocovariance. By Property 2.6, this means that $X_n$ and $Y_n$ have the same distribution. This
leads to the following property.

**Property 2.8. Reversibility of stationary Gaussian random processes** - Any
wide-sense stationary Gaussian random process is also strict-sense stationary
and reversible.

Finally, we note that all of these properties, strict-sense stationarity, wide-
sense stationarity, autocovariance, power spectral density, and reversibility
have natural generalizations in 2-D and higher dimensions. In particular, let $X_{s_1,s_2}$ be a 2-D discrete-space random process that is wide-sense stationary. Then its 2-D autocovariance function is given by

$$R(s_1 - r_1, s_2 - r_2) = \mathbb{E}[(X_{s_1,s_2} - \mu)(X_{r_1,r_2} - \mu)].$$

where $\mu = \mathbb{E}[X_{s_1,s_2}]$. Furthermore, the 2-D power spectral density of $X_{s_1,s_2}$ is given by

$$S_X(e^{j\omega_1}, e^{j\omega_2}) = \sum_{s_1=-\infty}^{\infty} \sum_{s_2=-\infty}^{\infty} R(s_1, s_2) e^{-j\omega_1 s_1 - j\omega_2 s_2} \quad (2.22)$$

where the right-hand side of this equation is known as the discrete-space Fourier transform (DSFT) of the 2-D autocovariance. In addition, the 2-D random process is said to be stationary if $X_{s_1,s_2}$ and $Y_{s_1,s_2} = X_{s_1-k_1,s_2-k_2}$ have the same distribution for all integers $k_1$ and $k_2$; and the 2-D random process is said to be reversible if $X_{s_1,s_2}$ and $Y_{s_1,s_2} = X_{-s_1,-s_2}$ have the same distribution. Again, it can be shown that 2-D zero-mean wide-sense stationary Gaussian random processes are always reversible and strict-sense stationary.
2.5 Chapter Problems

1. Let $Y$ be a random variable which is uniformly distributed on the interval $[0,1]$; so that it has a PDF of $p_y(y) = 1$ for $0 \leq y \leq 1$ and 0 otherwise. Calculate the CDF of $Y$ and use this to calculate both the CDF and PDF of $Z = Y^2$.

2. In this problem, we present a method for generating a random variable, $Y$, with any valid CDF specified by $F_y(t)$. To do this, let $X$ be a uniformly distributed random variable on the interval $[0,1]$, and let $Y = f(X)$ where we define the function

$$f(u) = \inf \{ t \in \mathbb{R} | F_y(t) \geq u \} .$$

Prove that in the general case, $Y$ has the desired CDF. (Hint: First show that the following two sets are equal, $(-\infty, F_y(t)] = \{ u \in \mathbb{R} : f(u) \leq t \}$.)

3. Give an example of three random variables, $X_1$, $X_2$, and $X_3$ such that for $k = 1, 2, 3$, $P\{X_k = 1\} = P\{X_k = -1\} = \frac{1}{2}$, and such $X_k$ and $X_j$ are independent for all $k \neq j$, but such that $X_1$, $X_2$, and $X_3$ are not jointly independent.

4. Let $X$ be a jointly Gaussian random vector, and let $A \in \mathbb{R}^{M \times N}$ be a rank $M$ matrix. Then prove that the vector $Y = AX$ is also jointly Gaussian.

5. Let $X \sim N(0, R)$ where $R$ is a $p \times p$ symmetric positive-definite matrix.
   a) Prove that if for all $i \neq j$, $E[X_iX_j] = 0$ (i.e., $X_i$ and $X_j$ are uncorrelated), then $X_i$ and $X_j$ are pair-wise independent.
   b) Prove that if for all $i, j$, $X_i$ and $X_j$ are uncorrelated, then the components of $X$ are jointly independent.

6. Let $X \sim N(0, R)$ where $R$ is a $p \times p$ symmetric positive-definite matrix. Further define the precision matrix, $B = R^{-1}$, and use the notation

$$B = \begin{bmatrix} 1/\sigma^2 & A \\ A^t & C \end{bmatrix} ,$$

where $A \in \mathbb{R}^{1 \times (p-1)}$ and $C \in \mathbb{R}^{(p-1) \times (p-1)}$. 
a) Calculate the marginal density of $X_1$, the first component of $X$, given the components of the matrix $R$.

b) Calculate the conditional density of $X_1$ given all the remaining components, $Y = [X_2, \cdots, X_p]^t$.

c) What is the conditional mean and covariance of $X_1$ given $Y$?

7. Let $X \sim N(0, R)$ where $R$ is a $p \times p$ symmetric positive-definite matrix with an eigen decomposition of the form $R = E \Lambda E^t$.

a) Calculate the covariance of $\tilde{X} = E^t X$, and show that the components of $\tilde{X}$ are jointly independent Gaussian random variables. (Hint: Use the result of problem 5 above.)

b) Show that if $Y = E \Lambda^{1/2} W$ where $W \sim N(0, I)$, then $Y \sim N(0, R)$. How can this result be of practical value?

8. For each of the following cost functions, find expressions for the minimum risk Bayesian estimator, and show that it minimizes the risk over all estimators.

a) $C(x, \hat{x}) = |x - \hat{x}|^2$.

b) $C(x, \hat{x}) = |x - \hat{x}|$

9. Let $\{Y_i\}_{i=1}^n$ be i.i.d. Bernoulli random variables with distribution

$$P\{Y_i = 1\} = \theta$$

$$P\{Y_i = 0\} = 1 - \theta$$

Compute the ML estimate of $\theta$.

10. Let $\{X_i\}_{i=1}^n$ be i.i.d. random variables with distribution

$$P\{X_i = k\} = \pi_k$$

where $\sum_{k=1}^m \pi_k = 1$. Compute the ML estimate of the parameter vector $\theta = [\pi_1, \cdots, \pi_m]$. (Hint: You may use the method of Lagrange multipliers to calculate the solution to the constrained optimization.)

11. Let $X_1, \cdots, X_n$ be i.i.d. random variables with distribution $N(\mu, \sigma^2)$. Calculate the ML estimate of the parameter vector $(\mu, \sigma^2)$. 
12. Let $X_1, \cdots, X_n$ be i.i.d. Gaussian random vectors with distribution $N(\mu, R)$ where $\mu \in \mathbb{R}^p$ and $R \in \mathbb{R}^{p \times p}$ is a symmetric positive-definite matrix, and let $X = [X_1, \cdots, X_n]$ be the $p \times n$ matrix containing all the random vectors. Let $\theta = [\mu, R]$ denote the parameter vector for the distribution.

a) Derive the expressions for the probability density of $p_x(x|\theta)$ with the forms given in equations (2.6) and (2.10). (Hint: Use the trace property of equation (2.7).)

b) Compute the joint ML estimate of $\mu$ and $R$.

13. Let $X$ and $W$ be independent Gaussian random vectors of dimension $p$ such that $X \sim N(0, R_x)$ and $W \sim N(0, R_w)$, and let $\theta$ be a deterministic vector of dimension $p$.

a) First assume that $Y = \theta + W$, and calculate the ML estimate of $\theta$ given $Y$.

b) For the next parts, assume that $Y = X + W$, and calculate an expression for $p_{x|y}(x|y)$, the conditional density of $X$ given $Y$.

c) Calculate the MMSE estimate of $X$ when $Y = X + W$.

d) Calculate an expression for the conditional variance of $X$ given $Y$.

14. Show that if $X$ and $Y$ are jointly Gaussian random vectors, then the conditional distribution of $X$ given $Y$ is also Gaussian.

15. Prove that if $X \in \mathbb{R}^M$ and $Y \in \mathbb{R}^N$ are zero-mean jointly Gaussian random vectors, then $E[X|Y] = AY$ and

$$E[(X - E[X|Y])(X - E[X|Y])^t|Y] = C,$$

where $A \in \mathbb{R}^{M \times N}$ and $C \in \mathbb{R}^{M \times M}$, i.e., Property 2.5.

16. Let $Y \in \mathbb{R}^M$ and $X \in \mathbb{R}^N$ be zero-mean jointly Gaussian random vectors. Then define the following notation for this problem. Let $p(y, x)$ and $p(y|x)$ be the joint and conditional density of $Y$ given $X$. Let $B$ be the joint positive-definite precision matrix (i.e., inverse covariance matrix) given by $B^{-1} = E[ZZ']$ where $Z = \begin{bmatrix} Y \\ X \end{bmatrix}$. Furthermore, let $C$, $D$, and $E$ be the matrix blocks that form $B$, so that

$$B = \begin{bmatrix} C & D \\ D^t & E \end{bmatrix}.$$
where \( C \in \mathbb{R}^{M \times M} \), \( D \in \mathbb{R}^{M \times N} \), and \( E \in \mathbb{R}^{N \times N} \). Finally, define the matrix \( A \) so that \( AX = \mathbb{E}[Y|X] \), and define the matrix

\[
\Lambda^{-1} = \mathbb{E}[(Y - \mathbb{E}[Y|X])(Y - \mathbb{E}[Y|X])^t|X].
\]

a) Write out an expression for \( p(y, x) \) in terms of \( B \).

b) Write out an expression for \( p(y|x) \) in terms of \( A \) and \( \Lambda \).

c) Derive an expression for \( \Lambda \) in terms of \( C \), \( D \), and \( E \).

d) Derive an expression for \( A \) in terms of \( C \), \( D \), and \( E \).

17. Let \( Y \) and \( X \) be random variables, and let \( Y_{MAP} \) and \( Y_{MMSE} \) be the MAP and MMSE estimates respectively of \( Y \) given \( X \). Pick distributions for \( Y \) and \( X \) so that the MAP estimator is very “poor”, but the MMSE estimator is “good”.

18. Prove that two zero-mean discrete-time Gaussian random processes have the same distribution, if and only if they have the same time autocovariance function.

19. Prove all Gaussian wide-sense stationary random processes are:
   a) strict-sense stationary
   b) reversible

20. Construct an example of a strict-sense stationary random process that is not reversible.

21. Consider two zero-mean Gaussian discrete-time random processes, \( X_n \) and \( Y_n \) related by

\[
Y_n = h_n * X_n,
\]

where \( * \) denotes discrete-time convolution and \( h_n \) is an impulse response with \( \sum_n |h_n| < \infty \). Show that

\[
R_y(n) = R_x(n) * h_n * h_{-n}.
\]
Chapter 3

Causal Gaussian Models

Perhaps the most basic tool in modeling is prediction. Intuitively, if one can effectively predict the behavior of something, then one must have an accurate model of its behavior. Clearly, an accurate model can enable accurate prediction, but we will demonstrate that the converse is also true: an accurate predictor can be used to create an accurate model.

In order to model data using prediction, we must decide the order in which prediction will occur. The simplest approach is to predict values in causal order, starting in the past and proceeding toward the future. In this chapter, we will show that causal prediction leads to many interesting and powerful tools; and, perhaps most importantly, it eliminates the possibility of circular dependencies in the prediction model.

However, the price we pay for causal prediction is that it requires that we impose a causal ordering on the data. For some types of data, causal ordering is quite natural. However, for images, which are the primary subject of this book, this is typically not the case, and imposing a causal order can often lead to artifacts in the results of processing.

3.1 Causal Prediction in Gaussian Models

Let \( X_1, X_2, \ldots, X_N \) be a portion of a discrete-time Gaussian random process. Without loss of generality, we will assume that \( X_n \) is zero-mean, since we may always subtract the mean in a preprocessing step. At any particular time \( n \), we may partition the random process into three distinct portions.

**The Past** - \( X_k \) for \( 1 \leq k < n \)
The Present - $X_n$

The Future - $X_k$ for $n < k \leq N$

Our objective is to predict the current value, $X_n$, from the past. As we saw in Chapter 2, one reasonable predictor is the MMSE estimate of $X_n$ given by

$$\hat{X}_n \triangleq \mathbb{E}[X_n | X_i \text{ for } i < n].$$

We will refer to $\hat{X}$ as a **causal predictor** since it only uses the past to predict the present value, and we define the **causal prediction error** as

$$\mathcal{E}_n = X_n - \hat{X}_n.$$

In order to simplify notation, let $\mathcal{F}_n$ denote the set of past observations given by $\mathcal{F}_n = \{X_i \text{ for } i < n\}$. Then the MMSE causal predictor of $X_n$ can be succinctly expressed as

$$\hat{X}_n = \mathbb{E}[X_n | \mathcal{F}_n].$$

Causal prediction leads to a number of very interesting and useful properties, the first of which is listed below.

**Property 3.1. Linearity of Gaussian predictor** - The MMSE causal predictor for a zero-mean Gaussian random process is a linear function of the past, i.e.,

$$\hat{X}_n = \mathbb{E}[X_n | \mathcal{F}_n] = \sum_{i=1}^{n-1} h_{n,i} X_i \quad (3.1)$$

where $h_{n,i}$ are scalar coefficients.

This property is a direct result of the linearity of conditional expectation for zero-mean Gaussian random vectors (Property 2.5). From this, we also know that the prediction errors must be a linear function of the past and present values of $X$.

$$\mathcal{E}_n = X_n - \sum_{i=1}^{n-1} h_{n,i} X_i \quad (3.2)$$

Another important property of causal prediction is that the prediction
error, $\mathcal{E}_n$, is uncorrelated from all past values of $X_i$ for $i < n$.

\[
\mathbb{E}[X_i \mathcal{E}_n] = \mathbb{E}[X_i (X_n - \hat{X}_n)] \\
= \mathbb{E}[X_i X_n] - \mathbb{E}[X_i \hat{X}_n] \\
= \mathbb{E}[X_i X_n] - \mathbb{E}[X_i \mathbb{E}[X_n | \mathcal{F}_n]] \\
= \mathbb{E}[X_i X_n] - \mathbb{E}[\mathbb{E}[X_i X_n | \mathcal{F}_n]] \\
= \mathbb{E}[X_i X_n] - \mathbb{E}[X_i X_n] \\
= 0
\]

Notice that the fourth equality is a result of Property 2.4 and the fifth equality is a result of Property 2.3. Since the combination of $\mathcal{E}_n$ and $X_0, \cdots, X_{n-1}$ are jointly Gaussian, this result implies that the prediction errors are independent of past values of $X$, which is stated in the following property.

**Property 3.2. Independence of causal Gaussian prediction errors from past**

The MMSE causal prediction errors for a zero-mean Gaussian random process are independent of the past of the random process. Formally, we write that for all $n$

\[
\mathcal{E}_n \perp \perp (X_1, \cdots, X_{n-1})
\]

where the symbol $\perp \perp$ indicates that the two quantities on the left and right are jointly independent of each other.

A similar approach can be used to compute the autocovariance between predictions errors themselves. If we assume that $i < n$, then the covariance between prediction errors is given by

\[
\mathbb{E}[\mathcal{E}_n \mathcal{E}_i] = \mathbb{E}
\left[
(X_n - \hat{X}_n) (X_i - \hat{X}_i)
\right]
\]

\[
= \mathbb{E}
\left[
(X_n - \mathbb{E}[X_n | \mathcal{F}_n]) (X_i - \mathbb{E}[X_i | \mathcal{F}_i])
\right]
\]

\[
= \mathbb{E}
\left[
\mathbb{E}
\left[
(X_n - \mathbb{E}[X_n | \mathcal{F}_n]) (X_i - \mathbb{E}[X_i | \mathcal{F}_i]) | \mathcal{F}_n
\right]
\right]
\]

\[
= \mathbb{E}
\left[
(X_i - \mathbb{E}[X_i | \mathcal{F}_i]) \mathbb{E}
\left[
(X_n - \mathbb{E}[X_n | \mathcal{F}_n]) | \mathcal{F}_n
\right]
\right]
\]

\[
= \mathbb{E}
\left[
(X_i - \mathbb{E}[X_i | \mathcal{F}_i]) (\mathbb{E}[X_n | \mathcal{F}_n] - \mathbb{E}[X_n | \mathcal{F}_n])
\right]
\]

\[
= \mathbb{E}[(X_i - \mathbb{E}[X_i | \mathcal{F}_i]) 0] = 0
\]

By symmetry, this result must also hold for $i > n$. Also, since we know that
the prediction errors are jointly Gaussian, we can therefore conclude joint independence from this result.

**Property 3.3. Joint independence of causal Gaussian prediction errors** - The MMSE causal prediction errors for a zero-mean Gaussian random process are jointly independent which implies that for all \( i \neq j \), \( \mathcal{E}_i \perp \perp \mathcal{E}_j \).

The causal prediction errors are independent, but we do not know their variance. So we denote the **causal prediction variance** as

\[
\sigma_n^2 \triangleq \mathbb{E}[\mathcal{E}_n^2].
\]

The prediction equations of (3.1) and (3.2) can be compactly expressed using vector-matrix notation. To do this, we let \( X, \hat{X}, \) and \( \mathcal{E} \) denote column vectors with elements indexed from 1 to \( N \). So for example, \( X = [X_1, \ldots, X_N]^t \).

Then the causal prediction equation of (3.1) becomes

\[
\hat{X} = HX
\]

where \( H \) is an \( N \times N \) causal prediction matrix containing the prediction coefficients, \( h_{i,j} \). By relating the entries of \( H \) to the coefficients of (3.1), we can see that \( H \) is a lower triangular matrix with zeros on the diagonal and with the following specific form.

\[
H = \begin{bmatrix}
0 & 0 & \cdots & 0 \\
h_{2,1} & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
h_{N-1,1} & h_{N-1,2} & \cdots & 0 & 0 \\
h_{N,1} & h_{N,2} & \cdots & h_{N,N-1} & 0
\end{bmatrix}
\]

Using this notation, the causal prediction error is then given by

\[
\mathcal{E} = (I - H)X = AX,
\]

where \( A = I - H \).

### 3.2 Density Functions Based on Causal Prediction

We can derive compact expressions for the density of both the prediction error, \( \mathcal{E} \), and the data, \( X \), by using the vector-matrix notation of the previous
section. To do this, first define $\Lambda = \text{diag} \{ \sigma_1^2, \cdots, \sigma_N^2 \}$ to be a diagonal matrix containing the causal prediction variances. Then the covariance of $\mathcal{E}$ is given by

$$\mathbb{E}[\mathcal{E}\mathcal{E}^t] = \Lambda$$

due to the independence of the prediction errors. Using the general form of the density function for a zero-mean multivariate Gaussian random vector, we then can write the density function for $\mathcal{E}$ as

$$p_\mathcal{E}(e) = \frac{1}{(2\pi)^{N/2} |\Lambda|^{-1/2}} \exp \left\{ -\frac{1}{2} e^t \Lambda^{-1} e \right\} .$$  \hspace{1cm} (3.5)$$

Since $\mathcal{E}$ and $X$ are related through a bijective transformation, it can be easily shown that the density of $X$ is proportional to the density of $\mathcal{E}$, with the absolute value of the Jacobian determinant of the transformation serving as the constant of proportionality. For this particular linear relationship between $X$ and $\mathcal{E}$, the probability densities are related by

$$p_x(x) = |\det(A)| \ p_\mathcal{E}(Ax)$$

where $|\det(A)|$ is the absolute value of the determinant of the matrix $A$. Fortunately, because $A$ is a causal predictor, it is constrained to be lower triangular with 1’s on its diagonal. Therefore, its determinant is one. Applying this result, and using the form of the density function for $p_\mathcal{E}(e)$ of (3.5) yields

$$p_x(x) = \frac{1}{(2\pi)^{N/2} |\Lambda|^{-1/2}} \exp \left\{ -\frac{1}{2} x^t A^t \Lambda^{-1} A x \right\} .$$  \hspace{1cm} (3.6)$$

From this, we can also see that the covariance of $X$ is given by

$$R_X = (A^t \Lambda^{-1} A)^{-1} ,$$

where $A$ is the causal prediction matrix and $\Lambda$ is the diagonal matrix of causal prediction variances.

### 3.3 1-D Gaussian Autoregressive (AR) Models

Time invariance is a very important concept that plays an essential role in the modeling of data. This is because in many practical cases it is reasonable to assume that the characteristic behavior of data does not change with time.
One method for enforcing time-invariance in a random processes is to specify that the parameters of the model do not change with time. When this is the case, we say that the model is **homogeneous**. So for example, a causal prediction model is homogeneous if the prediction filter and the prediction variance do not change with time. In this case, the MMSE causal predictor must be a linear time-invariant filter, so that the predictions are given by

$$\hat{X}_n = \sum_{i=1}^{N} h_{n-i} X_i,$$

where \( h_n \) is a **causal prediction filter** (i.e., \( h_n = 0 \) for \( n \leq 0 \)) and the causal prediction variances take on a constant value of \( \sigma_C^2 \).

When the prediction filter is time invariant, then the prediction matrices, \( H \) and \( A \), of equations (3.3) and (3.4) are said to be Toeplitz. A matrix is **Toeplitz** if there is a function, \( h_n \), so that \( H_{i,j} = h_{i-j} \). The structure of a Toeplitz matrix is illustrated in Figure 3.2. Intuitively, each row of a Toeplitz matrix is a shifted version of a single 1-D function, and multiplication by a Toeplitz matrix is essentially time-invariant convolution, but using truncated boundary conditions. Toeplitz matrices arise in many applications, and their spatial structure sometimes presents computational advantages.

In the same way that Toeplitz matrices arise from convolution with truncated boundaries, **circulant matrices** arise when convolution is implemented with circular boundary conditions. In this case, \( H_{i,j} = h_{(i-j) \ mod \ N} \) where \( mod \ N \) specifies the use of modulo \( N \) arithmetic. Multiplication by a circulant matrix, \( H \), is equivalent to circular convolution with the function \( h_n \). Circulant matrices have many useful properties because we know that circular convolution can be implemented with multiplication after taking the
3.3 1-D Gaussian Autoregressive (AR) Models

$$H = \begin{bmatrix}
h_0 & h_{-1} & \cdots & h_{2-N} & h_{1-N} \\
h_1 & h_0 & \cdots & h_{3-N} & h_{2-N} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
h_{N-2} & h_{N-3} & \cdots & h_0 & h_{-1} \\
h_{N-1} & h_{N-2} & \cdots & h_{1} & h_0
\end{bmatrix}$$

$$H = \begin{bmatrix}
h_0 & h_{N-1} & \cdots & h_{2} & h_{1} \\
h_1 & h_0 & \cdots & h_{3} & h_{2} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
h_{N-2} & h_{N-3} & \cdots & h_0 & h_{N-1} \\
h_{N-1} & h_{N-2} & \cdots & h_{1} & h_0
\end{bmatrix}$$

Figure 3.2: Diagram illustrating the structure of $N \times N$ Toeplitz and circulant matrices. Toeplitz matrices represent truncated convolution in time, and circulant matrices represent circular convolution in time.

discrete Fourier transform (DFT) of a signal.

A simple way to get around problems with boundary conditions is to extend the random process $X_n$ so that $n = -\infty, \cdots, -1, 0, 1, \cdots, \infty$. When the signal has infinite extent, then we can use the standard notation of linear time-invariant systems. In this case,

$$\mathcal{E}_n = X_n - \hat{X}_n$$
$$= X_n - X_n \ast h_n$$
$$= X_n \ast (\delta_n - h_n) ,$$

(3.7)

where $\ast$ denotes 1-D discrete-time convolution. Turning things around, we may also write

$$X_n = \mathcal{E}_n + X_n \ast h_n$$
$$= \mathcal{E}_n + \sum_{i=1}^{P} X_{n-i} h_i ,$$

(3.8)

which is the form of a $P^{th}$ order infinite impulse response (IIR) filter.

When $P = \infty$, then all past values of $X_n$ can be used in the MMSE prediction. However if $P < \infty$, then the prediction only depends on the last $P$ observations, and we call $X_n$ an autoregressive (AR) random process. Figure 3.3 shows how the predictor in an order $P$ AR model only depends on the $P$ past neighbors.

Equation (3.8) is also sometimes called a white noise driven model for the AR process because it has the form of an linear time-invariant (LTI) system with a white noise input, $\mathcal{E}_n$. The white noise driven model is
particularly useful because it provides an easy method for generating an AR process, $X_n$. To do this, one simply generates a sequence of i.i.d. Gaussian random variables with distribution $N(0, \sigma_C^2)$, and filters them with the IIR filter of equation (3.8). If the IIR filter is stable, then its output, $X_n$, will be a stationary random process.

We can calculate the autocovariance of the AR process, $X_n$, by using the relationship of equation (3.8). Since the prediction errors, $\mathcal{E}_n$, are i.i.d., we know that their time autocovariance is given by

$$R_{\mathcal{E}}(i-j) = \mathbb{E}[\mathcal{E}_i\mathcal{E}_j] = \sigma_C^2 \delta_{i-j}.$$  

From the results of Section 2.4 (See problem 21 of Chapter 2), and the relationship of equation (3.7), we know that the autocovariance of $X_n$ obeys the following important relationship

$$R_X(n) \ast (\delta_n - h_n) \ast (\delta_n - h_{-n}) = R_{\mathcal{E}}(n) = \sigma_C^2 \delta_n . \quad (3.9)$$

Taking the DTFT of the time autocovariance, we then get an expression for the power spectral density of the AR process.

$$S_X(\omega) = \frac{\sigma_C^2}{|1 - H(\omega)|^2} . \quad (3.10)$$

**Example 3.1.** Consider the $P^{th}$ order AR random process $X_1, \cdots, X_N$ with prediction variance of $\sigma_C^2$, and prediction errors given by

$$\mathcal{E}_n = X_n - \sum_{i=1}^P h_i X_{n-i} . \quad (3.11)$$

To simplify notation, we will assume that $X_n = 0$ when $n < 0$, so that we do not need to use special indexing at the boundaries of the signal.

Our task in this example is to compute the joint ML estimate of the prediction filter, $h_n$, and the prediction variance, $\sigma_C^2$. To do this, we first must compute the probability density of $X$. Using the PDF of equation (3.6), we can write the density for the AR process as

$$p(x) = \frac{1}{(2\pi \sigma_C^2)^{N/2}} \exp \left\{ -\frac{1}{2\sigma_C^2} \sum_{n=1}^N \left( x_n - \sum_{i=1}^P h_i x_{n-i} \right)^2 \right\} . \quad (3.6)$$
We can further simplify the expression by defining the following parameter vector and statistic vector.

\[
\begin{align*}
  h &= [h_1, h_2, \cdots, h_P]^t \\
  Z_n &= [X_{n-1}, X_{n-2}, \cdots, X_{n-P}]^t
\end{align*}
\]

Then the log likelihood of the observations, \( X \), can be written as

\[
\log p(X) = -\frac{N}{2} \log (2\pi \sigma_C^2) - \frac{1}{2\sigma_C^2} \sum_{n=1}^{N} \left( X_n - \sum_{i=1}^{P} h_i X_{n-i} \right)^2
\]

Using this, we can express the log likelihood as

\[
\log p(X) = -\frac{N}{2} \log (2\pi \sigma_C^2) - \frac{1}{2\sigma_C^2} \sum_{n=1}^{N} (X_n^2 - 2h^t Z_n X_n + h^t Z_n Z_n^t h)
\]

where

\[
\begin{align*}
  \hat{\sigma}_x^2 &= \frac{1}{N} \sum_{n=1}^{N} X_n^2 \\
  \hat{b} &= \frac{1}{N} \sum_{n=1}^{N} Z_n X_n \\
  \hat{R} &= \frac{1}{N} \sum_{n=1}^{N} Z_n Z_n^t
\end{align*}
\]

Notice that the three quantities \( \hat{\sigma}_x^2 \), \( \hat{b} \), and \( \hat{R} \) are sample statistics computed from the data, \( X \). Intuitively, they correspond to estimates of the variance of \( X_n \), the covariance between \( Z_n \) and \( X_n \), and the autocovariance of \( Z_n \).

First, we compute the ML estimate of the prediction filter by taking the gradient with respect to the filter vector \( h \).

\[
\nabla_h \log p(X) = -\frac{N}{2\sigma_C^2} \nabla_h \left( h^t \hat{R} h - 2h^t \hat{b} \right)
\]

\[
= -\frac{N}{\sigma_C^2} \left( \hat{R} h - \hat{b} \right),
\]
where we use the convention that the gradient is represented as a column vector. Setting the gradient of the log likelihood to zero, results in the ML estimate of the prediction filter, $h$.

$$\hat{h} = \hat{R}^{-1}\hat{b}$$

We can next compute the ML estimate of $\sigma_C^2$ by plugging in the expression for the ML estimate of $h$, and differentiating with respect to the parameter $\sigma_C^2$.

$$\frac{d}{d\sigma_C^2} \log p(X) = \frac{d}{d\sigma_C^2} \left[ -\frac{N}{2} \log (2\pi \sigma_C^2) - \frac{N}{2\sigma_C^2} \left( \hat{\sigma}_x^2 - \hat{b}^t \hat{R}^{-1} \hat{b} \right) \right]$$

$$= -\frac{N}{2} \left[ \frac{1}{\sigma_C^2} - \frac{1}{\sigma_C^4} \left( \hat{\sigma}_x^2 - \hat{b}^t \hat{R}^{-1} \hat{b} \right) \right].$$

Setting the derivative of the log likelihood to zero, results in the expression

$$1 - \frac{1}{\hat{\sigma}_C^2} \left( \hat{\sigma}_x^2 - \hat{b}^t \hat{R}^{-1} \hat{b} \right) = 0$$

Which yields the ML estimate of the causal prediction variance given by

$$\hat{\sigma}_C^2 = \hat{\sigma}_x^2 - \hat{b}^t \hat{R}^{-1} \hat{b}.$$

Perhaps a more intuitive representation for $\hat{\sigma}_C^2$ is as the average of the squared prediction errors.

$$\hat{\sigma}_C^2 = \frac{1}{N} \sum_{n=1}^{N} \left( X_n - \hat{h}^t Z_n \right)^2$$

$$= \frac{1}{N} \sum_{n=1}^{N} \varepsilon_n^2$$

### 3.4 2-D Gaussian AR Models

In fact, the analysis of 1-D AR models from the previous sections is easily generalized to a regular grid in 2 or more dimensions. To do this, each lattice point is represented by $s = (s_1, s_2)$, where $s$ is a vector index with
3.4 2-D Gaussian AR Models

The pixel denoted by the symbol $\otimes$ is predicted using the past values denoted by the symbol $\bullet$. For the 1-D case, $P = 2$ past values are used by the predictor; and for the 2-D case, $2P(P+1) = 12$ past values are used by the predictor. Notice that the asymmetric shape of the 2-D prediction window results from raster ordering of the pixels.

Each coordinate taking on values in the range 1 to $N$. When visualizing 2-D models, we will generally assume that $s_1$ is the row index of the data array and $s_2$ is the column index.

The key issue in generalizing the AR model to 2-D is the ordering of the points in the plane. Of course, there is no truly natural ordering of the points, but a common choice is raster ordering, going left to right and top to bottom in much the same way that one reads a page of English text. Using this ordering, the pixels of the image, $X_s$, may be arranged into a vector with the form

$$X = [X_{1,1}, \ldots, X_{1,N}, X_{2,1}, \ldots, X_{2,N}, \ldots, X_{N,1}, \ldots, X_{N,N}]^t,$$

and the causal prediction errors, $\mathcal{E}$, may be similarly ordered. If we assume that $X(s_1, s_2) = 0$ outside the range of $1 \leq s_1 \leq N$ and $1 \leq s_2 \leq N$, then the 2-D causal prediction error can again be written as

$$\mathcal{E}_s = X_s - \sum_{r \in W_P} h_r X_{s-r},$$

where $W_P$ is a window of past pixels in 2-D. Typically, this set is given by

$$W_P = \left\{ r = (r_1, r_2) : \begin{cases} (r_1 = 0 \text{ and } 1 \leq r_2 \leq P) \\ (1 \leq r_1 \leq P \text{ and } -P \leq r_2 \leq P) \end{cases} \right\}. \quad (3.12)$$

Notice that using this definition of $W$, the prediction sum only contains previous pixels of $X_s$ in raster order. The resulting asymmetric window shown in Figure 3.3 contains a total of $2P(P+1)$ pixels. The window is not symmetric because it is constrained by the raster ordering.
Using this convention, the prediction errors can be expressed in matrix form as

$$\mathbf{E} = (I - H) \mathbf{X},$$

where $H$ is the 2-D prediction matrix.

Since $H$ represents the application of a linear space-invariant 2-D filter, it has a special structure and is referred to as a **Toeplitz block Toeplitz** matrix. Figure 3.4 illustrates the structure graphically. Notice that the matrix $H$ is formed by a set of blocks, $B_k$, organized in a Toeplitz structure. In addition, each individual block is itself a Toeplitz matrix, which explains the terminology. For our problem each block is of size $N \times N$; so in this case, the Toeplitz block Toeplitz structure implies that the matrix, $H_{i,j}$, can be expressed in the form

$$H_{mN+k,nN+l} = h_{(m-n,k-l)}$$

where $h_{(i,j)}$ is the 2-D prediction filter’s impulse response. Intuitively, a Toeplitz block Toeplitz matrix results whenever space-invariant filters are represented as matrix operators and the pixels in the image are organized in raster order.

The 2-D AR model also has properties quite similar to the 1-D case. In fact, the results are formally identical, only with 1-D convolution being replaced by 2-D convolution and the 1-D DTFT being replaced by the 2-D DSFT. More specifically,

$$R_X(s) * (\delta_s - h_s) * (\delta_s - h_{-s}) = \sigma_C^2 \delta_s, \quad (3.13)$$
where * denotes 2-D convolution of 2-D functions, and

$$S_X(\mu, \nu) = \frac{\sigma_C^2}{|1 - H(\mu, \nu)|^2}.$$  \hspace{1cm} (3.14)

where $H(\mu, \nu)$ is the DSFT of $h_s$, and $S_X(\mu, \nu)$ is the 2-D power spectral density of the AR process. (See Section 2.4 for details of the 2-D power spectrum.)

**Example 3.2.** Consider the $P^{th}$ order 2-D AR random process $X_s$ for $s \in S = \{(s_1, s_2) : 1 \leq s_1, s_2 \leq N\}$ with prediction variance of $\sigma_C^2$ and prediction errors given by

$$\mathcal{E}_s = X_s - \sum_{r \in W} h_r X_{s-r}.$$  \hspace{1cm} (3.15)

As before, we will assume that $X_s = 0$ when $s_1 < 0$ or $s_2 < 0$. Following along the lines of Example 3.1, we can compute the probability density of $X$ using the PDF of equation (3.6).

$$p(x) = \frac{1}{(2\pi\sigma_C^2)^{N^2/2}} \exp \left\{ -\frac{1}{2\sigma_C^2} \sum_{s \in S} \left( x_s - \sum_{r \in W} h_r x_{s-r} \right)^2 \right\}.$$  

In 2-D, we can define the following parameter vector and statistic vector.

$$h = [h_{(0,1)}, \cdots, h_{(0,P)}, h_{(1,-P)}, \cdots, h_{(1,P)}, \cdots, h_{(P,P)}]^t$$

$$Z_s = [X_{s-(0,1)}, \cdots, X_{s-(0,P)}, X_{s-(1,-P)}, \cdots, X_{s-(1,P)}, \cdots, X_{s-(P,P)}]^t.$$  

Using these definitions, the prediction error can be compactly written as

$$\mathcal{E}_s = X_s - h^t Z_s,$$

and the log likelihood can be written as

$$\log p(X) = -\frac{N^2}{2} \log (2\pi\sigma_C^2) - \frac{N^2}{2\sigma_C^2} \left( \hat{\sigma}_x^2 - 2h^t \hat{b} + h^t \hat{R} h \right).$$

In 2-D, we can define the following parameter vector and statistic vector.
where

\[ \hat{\sigma}^2_x = \frac{1}{N^2} \sum_{s \in S} X_s^2 \]
\[ \hat{b} = \frac{1}{N^2} \sum_{s \in S} Z_s X_s \]
\[ \hat{R} = \frac{1}{N^2} \sum_{s \in S} Z_s Z_s^t . \]

A calculation similar to that in Example 3.1 then results in the ML estimates of the model parameters \( h \) and \( \sigma^2_C \).

\[ \hat{h} = \hat{R}^{-1} \hat{b} \]
\[ \hat{\sigma}^2_C = \hat{\sigma}^2_x - \hat{b}' \hat{R}^{-1} \hat{b} , \]

or equivalently

\[ \hat{\sigma}^2_C = \frac{1}{N^2} \sum_{s \in S} \left( X_s - \hat{h}' Z_s \right)^2 . \]
3.5 Chapter Problems

1. Let \( \{X_n\}_{n=1}^N \) be a 1-D Gaussian random process such that

\[
\mathcal{E}_n = X_n - \sum_{i=n-P}^{n-1} h_{n-i} X_i
\]

results in \( \mathcal{E}_n \) being a sequence of i.i.d. \( N(0, \sigma^2) \) random variables for \( n = 1, \cdots, N \), and assume that \( X_n = 0 \) for \( n \leq 0 \). Compute the ML estimates of the prediction filter \( h_n \) and the prediction variance \( \sigma^2 \).

2. Let \( X_n \) be samples of a Gaussian AR process with order \( P \) and parameters \( (\sigma^2, h) \). Also make the assumption that \( X_n = 0 \) for \( n \leq 0 \).

   a) Use Matlab to generate 100 samples of \( X \). Experiment with a variety of values for \( P \) and \( (\sigma^2, h) \). Plot your output for each experiment.

   b) Use your sample values of \( X \) generated in part a) to compute the ML estimates of the \( (\sigma^2, h) \), and compare them to the true values.

3. Let \( X \) be a Gaussian 1-D AR process with \( h_n = \rho \delta_{n-1} \) and prediction variance \( \sigma^2 \).

   a) Analytically calculate \( S_X(\omega) \), the power spectrum of \( X \), and \( R_X(n) \), the autocovariance function for \( X \).

   b) Plot \( S_X(\omega) \) and \( R_X(n) \) for \( \rho = 0.5 \) and \( \rho = 0.95 \).

4. Let \( X_n \) be a zero-mean wide-sense stationary random process, and define

\[
Z_n = \begin{bmatrix} X_{n-1} \\ \vdots \\ X_{n-P} \end{bmatrix}
\]

for some fixed order \( P \), and let

\[
R = \mathbb{E} \left[ \frac{1}{N} \sum_{n=0}^{N-1} Z_n Z_n^t \right]
\]

   a) Show that \( R \) is a Toeplitz matrix.

   b) Show that \( R \) is a positive semi-definite matrix.
5. Consider an LTI system with input \( x_n \), output \( y_n \), and impulse response, \( h_n \), so that \( y_n = h_n \ast x_n \), where \( \ast \) denotes convolution. Also define the vectors, \( y = [y_0, \cdots, y_{N-1}]^t \), and \( x = [x_0, \cdots, x_{N-1}]^t \). Show that if \( x_n = 0 \) for \( n < 0 \) and \( n \geq N \), then

\[
y = Ax
\]

where \( A \) is a Toeplitz matrix.

6. Consider a linear system with input \( \{x_n\}_{n=0}^{N-1} \), output, \( \{y_n\}_{n=0}^{N-1} \), and impulse response, \( \{h_n\}_{n=0}^{N-1} \), so that \( y_n = h_n \otimes x_n \), where \( \otimes \) denotes circular convolution. Also define the vectors, \( y = [y_0, \cdots, y_{N-1}]^t \), and \( x = [x_0, \cdots, x_{N-1}]^t \). Show that if \( x_n = 0 \) for \( n \leq 0 \) and \( n > N \), then

\[
y = Ax
\]

where \( A \) is a circulant matrix.

7. Let \( A \) be an \( N \times N \) circulant matrix, so that \( A_{i,j} = h_{(i-j) \mod N} \), for some real-valued function \( h_n \). In order to simplify notation, we will assume all matrices in this problem are indexed from 0 to \( N - 1 \), rather than from 1 to \( N \). Using this convention, define the following matrix for \( 0 \leq m, n < N \),

\[
T_{m,n} = \frac{1}{\sqrt{N}} e^{-j \frac{2\pi mn}{N}}.
\]

Then \( T \) is known as the \( N \) dimensional orthonormal discrete Fourier transform (DFT)\(^1\)

a) Show that the DFT is an orthonormal transform by showing that the columns of the matrix are orthogonal and normal. So formally this means for \( 0 \leq m, k < N \)

\[
\sum_{n=0}^{N-1} T_{m,n} T_{k,n}^* = \delta_{m-k}.
\]

b) Show that inverse transformation is given by

\[
[T^{-1}]_{m,n} = T_{n,m}^*
\]

\(^1\)The DFT is conventionally defined without the factor of \( \frac{1}{\sqrt{N}} \), but we add this constant to normalize the transform.
3.5 Chapter Problems

where $T^{-1}$ is the inverse DFT.

c) Show that $\Lambda = TAT^{-1}$ is a diagonal matrix with entries given by the DFT of the function $h_n$. That is $\Lambda = \text{diag}\{\lambda_0, \cdots, \lambda_{N-1}\}$ where

$$\lambda_m = \sqrt{N} \sum_{n=0}^{N-1} T_{m,n} h_n.$$ 

d) Show that the eigenvalues of $A$ are the diagonal entries of $\Lambda$ and that the eigenvectors are the corresponding columns of $T^{-1}$.

e) Show that the logarithm of the absolute value of the determinant of the matrix $A$ is given by

$$\log |A| = \sum_{n=0}^{N-1} \log |\lambda_n|. $$

where $|A|$ denotes the absolute value of the determinant of $A$.

f) Show that in the limit at $N \to \infty$,

$$\lim_{N \to \infty} \frac{1}{N} \log |A| = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log |H(\omega)| d\omega. $$

where $H(\omega) = \sum_{n=0}^{\infty} h_n e^{-j\omega n}$ is the DTFT of $h_n$.

8. Let $X_{m,n}$ be a zero-mean 2-D AR Gaussian random process with $h_{m,n} = \rho \delta_{m-1} \delta_n + \rho \delta_m \delta_{n-1} - \rho^2 \delta_{m-1} \delta_{n-1}$ and prediction variance $\sigma^2$.

a) Calculate an expression for $E[|X_{m,n}|^2]$, and determine the value of $\sigma^2$ so that $E[|X_{m,n}|^2] = 1$.

b) Analytically calculate $S_X(\mu, \nu)$, the power spectrum of $X$.

c) Use Matlab to generate a $512 \times 512$ sample of $X$. Use the value of $\sigma^2$ from a) so that $E[|X_{m,n}|^2] = 1$, and use $\rho = 0.9$.

d) Use the Matlab command `imagesc()` to display $X$ as an image.

e) Repeat part c) for $\rho = 0.5$ and $\rho = 0.98$.

f) Plot $S_X(\mu, \nu)$ for $\rho = 0.9$.

9. Consider a 2-D LTI system with input $x_{m,n}$, output, $y_{m,n}$, and impulse response, $h_{m,n}$, so that $y_{m,n} = h_{m,n} \ast x_{m,n}$, where $\ast$ denotes 2-D convolution. Also define the vectors, $y = [y_{1,1}, \cdots, y_{1,N}, y_{2,1}, \cdots, y_{2,N}, \cdots, y_{N,1}, \cdots, y_{N,N}]^t$, ...
and \( x = [x_{1,1}, \cdots, x_{1,N}, x_{2,1}, \cdots, x_{2,N}, \cdots, x_{N,1}, \cdots, x_{N,N}]^t \). Show that if \( x_{m,n} = 0 \) for \( m < 1 \) or \( m > N \) or \( n < 1 \) or \( n > N \), then

\[
y = Ax
\]

where \( A \) is a Toeplitz block Toeplitz matrix.

10. The following problem builds on the results of Example 3.2. Let \( X_s \) be a zero-mean stationary 2-D Gaussian random process and let \( S = \{ s = (s_1, s_2) : 1 \leq s_1, s_2, \leq N \} \). Define, the augmented vector

\[
V_s = \begin{bmatrix} X_s \\ Z_s \end{bmatrix}
\]

where \( Z_s \) is as defined in Example 3.2. Then we can defined an augmented covariance matrix as

\[
\hat{C} \triangleq \frac{1}{N^2} \sum_{s \in S} V_s V_s^t = \begin{bmatrix} \hat{\sigma}^2_x & \hat{b}^t \\ \hat{b} & \hat{R} \end{bmatrix}.
\]

where, again, \( \hat{\sigma}^2_x \), \( \hat{b} \), and \( \hat{R} \) are defined as in Example 3.2. Furthermore, define the expected covariance matrix

\[
C \triangleq \mathbb{E} \left[ \hat{C} \right]
\]

a) Show that the MMSE causal prediction filter, \( h \), and prediction variance, \( \sigma_C^2 \), can be computed from the entries of the matrix \( C \); and determine an explicit expressions for \( h \) from the entries of \( C \).

b) Show \( C \) is a matrix with Toeplitz blocks.

c) Show that the matrix \( C \) can be specified by knowing the values of the space autocovariance function \( R(k, l) = \mathbb{E} \left[ X_{(m,n)}X_{(m-k,n-l)} \right] \).

d) Specify exactly the set of values of \( (k, l) \) for which you need to determine \( R(k, l) \) in order to construct the Toeplitz block Toeplitz matrix \( C \) for \( P = 2 \).

e) Remembering that \( R(k, l) = R(-k, -l) \), determine the number of distinct autocovariance values that are needed to construct the matrix \( C \) for \( P = 2 \).
f) Determine the number of parameters in a 2-D AR model of order $P = 2$.

g) Is the number of distinct autocovariance values determined in part e) equal to the number of AR model parameters in part f)?

h) Use the same methodology to determine the number of distinct autocovariance values and the number of AR model parameters for a 1-D AR process of order $P = 2$. Are these numbers equal?

i) What are the possible consequences of the mismatch in the number of model parameters and number of distinct autocovariance values for a 2-D AR model?
One disadvantage of AR processes is that their construction depends on a causal ordering of points in time. For many applications, it is completely reasonable to order points into the future, present, and past. For example, in real-time processing of audio signals, this is a very natural organization of the data. But sometimes, measurements have no natural ordering. For example, the temperatures measured along a road, are a 1-D signal, but the direction of causality is not well defined. Which end of the road should represent the past and which the future?

While this example may seem a bit contrived, the problem of ordering points becomes much more severe for pixels in an image. In practical imaging applications, such as video communications, it is often necessary to impose a raster ordering on pixels in an image, but subsequent 1-D processing of the ordered pixels is likely to produce artifacts aligned with the raster pattern.

The problem becomes even worse when the collected data has no regular structure. For example, the height of each person in an organization can be viewed as a random quantity to be modeled, but what 1-D ordering of people in an organization is appropriate? In this case, the set of data points does not even naturally lie on a 2-D or higher dimensional grid.

The objective of this section is to introduce the basic tools that we will need to remove causality from the modeling of images and other data. For images, these techniques will help us to avoid the creation of artifacts; but more generally, these techniques can serve as powerful tools for the modeling of a wide variety of data that can be represented on regular grids or graphs.
4.1 Non-Causal Prediction in Gaussian Models

In order to introduce the concepts of modeling with non-causal prediction, we will start with the case of 1-D signals. Let $X_1, \cdots, X_N$ again be a zero-mean discrete-time Gaussian random process. Rather than use causal prediction, we will attempt to model $X_n$ using predictions based on a combination of past and future information. In this case, the MMSE predictor is given by

$$\hat{X}_n = \mathbb{E}[X_n | X_i \text{ for } i \neq n] .$$

As with the causal predictor, the non-causal predictor is a linear function of the data when the random process is zero-mean and Gaussian. So the non-causal prediction error can be written as

$$\mathcal{E}_n = X_n - \sum_{i=1}^{N} g_{n,i} X_i$$

where $g_{n,n} = 0$ for all $1 \leq n \leq N$. This condition that $g_{n,n} = 0$ is very important. Otherwise, the value $X_n$ would be used to predict $X_n$ perfectly!

In addition, we define $\sigma_n^2$ to be the non-causal prediction variance given by

$$\sigma_n^2 = \mathbb{E}[\mathcal{E}_n^2 | X_i \text{ for } i \neq n] .$$

Notice that because $X_n$ is jointly Gaussian, we know from Property [2.5] that the prediction variance is not a function of $X_i$ for $i \neq n$; however, it may depend on the time index, $n$.

As in the case of causal predictors, non-causal predictors have a number of important properties. First, the non-causal prediction errors are independent of the values used in prediction.
Property 4.1. Independence of non-causal Gaussian prediction errors from past and future - The MMSE non-causal prediction errors for a zero-mean Gaussian random process are independent of the past and future of the random process.

\[ \mathcal{E}_n \perp \{X_i\}_{i \neq n} . \]

Unfortunately, one very important property is lost when using non-causal prediction. In general, the non-causal prediction errors are no longer uncorrelated and independent. This is a great loss because the independence of the causal prediction errors was essential for the computation of the probability density of the random process in Section 3.2.

4.2 Density Functions Based on Non-Causal Prediction

Although the non-causal prediction errors are not independent, we will still be able to calculate the probability density for \( X_n \) by using vector-matrix operations, but with a somewhat different strategy. Since \( X \) is a zero-mean Gaussian random vector, it must have a density function with the form

\[
p(x) = \frac{1}{(2\pi)^{N/2}|B|^{1/2}} \exp \left\{ -\frac{1}{2}x^t B x \right\} ,
\]

where \( B \) is the inverse of the covariance matrix for \( X \). The matrix \( B \) is often called the precision matrix since as it grows large, the variance of \( X \) tends to become small. Furthermore, we know that since \( X \) is a zero-mean Gaussian vector, we can write the conditional distribution of \( X_n \) given all the remaining \( X_i \) for \( i \neq n \) as

\[
p(x_n|x_i \text{ for } i \neq n) = \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp \left\{ -\frac{1}{2\sigma_n^2} \left( x_n - \sum_{i=1}^{N} g_{n,i} x_i \right)^2 \right\}
\]

where \( \sigma_n^2 \) is the non-causal prediction variance for \( X_n \), and \( g_{n,i} \) are the MMSE non-causal prediction coefficients.

Our objective is then to determine the matrix \( B \) in terms of the non-causal prediction parameters \( \sigma_n^2 \) and \( g_{n,i} \). We can derive this relationship by
differentiating the log likelihood as follows.

$$\frac{d}{dx_n} \log p(x) = \frac{d}{dx_n} \log \{ p(x_n|x_i \text{ for } i \neq n) p(x_i \text{ for } i \neq n) \}$$

$$= \frac{d}{dx_n} \log p(x_n|x_i \text{ for } i \neq n) + \frac{d}{dx_n} \log p(x_i \text{ for } i \neq n)$$

Next, we can evaluate the left and right hand sides of this equality by differentiating the expressions of (4.1) and (4.2) to yield the following new relationship.

$$\sum_{i=1}^{N} B_{n,i} x_i = \frac{1}{\sigma_n^2} \left( x_n - \sum_{i=1}^{N} g_{n,i} x_i \right)$$

Since this equality must hold for all $x$ and all $n$, the inverse covariance matrix, $B$, must then be given by

$$B_{i,j} = \frac{1}{\sigma_i^2} (\delta_{i-j} - g_{i,j}) .$$

Alternatively, we can invert this relationship to compute the MMSE non-causal predictor parameters given a specification of $B$.

$$\sigma_n^2 = (B_{n,n})^{-1}$$

$$g_{n,i} = \delta_{n-i} - \sigma_n^2 B_{n,i} .$$

These two relationships can be more compactly represented using matrix notation. If we define the matrix $G_{i,j} = g_{i,j}$ as the non-causal prediction matrix, $\Gamma = \text{diag} \{ \sigma_1^2, \cdots, \sigma_N^2 \}$ as the diagonal matrix of non-causal prediction variances, and $\mathcal{E}$ as the column vector of non-causal prediction errors, then we have that $\mathcal{E} = (I - G)X$, and

$$B = \Gamma^{-1}(I - G)$$

or alternatively

$$\Gamma = \text{diag}(B)^{-1}$$

$$G = I - \Gamma B .$$
4.3 1-D Gaussian Markov Random Fields (GMRF)

An important special case occurs when the number of observations needed to determine the MMSE non-causal predictor is limited to a window of \( n \pm P \) about the point being predicted. In order to simplify the notation, we define the window

\[ \partial n = \{ i \in [1, \cdots, N] : i \neq n \text{ and } |i - n| \leq P \} , \]

so that \( \partial n \) is a set containing \( P \) neighbors on either side of \( n \), except on the boundary, where it is truncated.

Using this new notation, a 1-D Gaussian Markov random field (GMRF)\(^1\) is any Gaussian random process with the property that

\[ \mathbb{E}[X_n|X_i \text{ for } i \neq n] = \mathbb{E}[X_n|X_i \text{ for } i \in \partial n] . \]

In words, the MMSE non-causal predictor for any pixel in an GMRF is only dependent on the pixel’s neighbors. Figure 4.2 illustrates the structure of the prediction window for a 1-D GMRF of order \( P \).

The GMRF is a bit like an AR process, but the causal predictor of the AR process is replaced with a non-causal predictor in the GMRF. For the most general case of a zero-mean GMRF, the non-causal prediction error then has the form

\[ \mathcal{E}_n = X_n - \sum_{i \in \partial n} g_{n,i} X_i . \]

In order to reduce the number of model parameters, it is often useful to assume that the prediction coefficients are not a function of position, \( n \). This results in the following new definition.

**Definition 5. Homogeneous GMRF**

A GMRF is said to be **homogeneous** if both the MMSE non-causal predictor and MMSE prediction variance are invariant to position.

The MMSE non-causal prediction error for a homogeneous GMRF then

\(^1\) We should note that the terminology “1-D random field” is clearly an oxymoron, since the term “random field” refers to a 2-D object. Later we will see that the concept of a Markov random field (MRF) grew out of the study of 2 or more dimensional objects, where they are most useful, but the concept applies perfectly well to 1-D also.
has the form
\[ E_n = X_n - \sum_{i \in \partial n} g_{n-i} X_i , \]
with the non-causal prediction variance, \( \sigma^2_{\text{NC}} \), taking on a constant value. In this case, we can write the density function for the homogeneous GMRF as
\[ p(x) = \frac{1}{(2\pi)^{N/2}|B|^{1/2}} \exp \left\{ -\frac{1}{2} x^t B x \right\}, \]
where
\[ B_{i,j} = \frac{1}{\sigma^2_{\text{NC}}} (\delta_{i-j} - g_{i-j}) . \]
Since we know that \( B \) must be a symmetric matrix, this implies that \( g_n = g_{-n} \) must be a symmetric filter.

Once again, if we extend \( X_n \) so that \( n = -\infty, \cdots, -1, 0, 1, \cdots, \infty \), then we can express the relation between \( X_n \) and the non-causal prediction errors, \( E_n \), using convolution.
\[ E_n = X_n * (\delta_n - g_n) \quad (4.3) \]
and
\[ X_n = E_n + \sum_{i \in \partial n} g_{n-i} X_i . \]
Using this expression, we can compute the covariance between \( E_n \) and \( X_n \) as
\[ \mathbb{E}[E_n X_n] = \mathbb{E} \left[ E_n \left( E_n + \sum_{i \in \partial n} g_{n-i} X_i \right) \right] = \mathbb{E}[E_n^2] + \sum_{i \in \partial n} g_{n-i} \mathbb{E}[E_n X_i] = \mathbb{E}[E_n^2] + 0 = \sigma^2_{\text{NC}} . \]
By combining this result with Property 4.1, we get the following expression for the covariance between the prediction error and \( X_n \).
\[ \mathbb{E}[E_n X_{n+k}] = \sigma^2_{\text{NC}} \delta_k . \]
This result just indicates that the prediction errors are independent of the
values used in the prediction. Using this fact, we have that

\[ \sigma_{NC}^2\delta_k = \mathbb{E}[\mathcal{E}_nX_{n+k}] \]

\[ = \mathbb{E}\left[\mathcal{E}_n\left(\mathcal{E}_{n+k} + \sum_{i \in \partial(n+k)} g_{n+k-i}X_i\right)\right] \]

\[ = \mathbb{E}[\mathcal{E}_n\mathcal{E}_{n+k}] + \sum_{i \in \partial(n+k)} g_{n+k-i}\mathbb{E}[\mathcal{E}_nX_i] \]

\[ = R_\xi(k) + \sum_{i \in \partial(n+k)} g_{n+k-i}\sigma_{NC}^2\delta_{i-n} \]

\[ = R_\xi(k) + \sigma_{NC}^2g_k. \]

Rearranging terms results in

\[ R_\xi(n) = \sigma_{NC}^2(\delta_n - g_n), \quad (4.4) \]

where \( R_\xi(n) \) is the time autocovariance of the non-causal prediction errors. So from this we see that, in general, the noncausal prediction errors are not white. From equation (4.3) and problem 21 of Chapter 2, we know that autocovariance functions for \( \mathcal{E}_n \) and \( X_n \) must be related by

\[ R_\xi(n) = R_X(n) \ast (\delta_n - g_n) \ast (\delta_n - g_{-n}). \tag{4.5} \]

Equating the expressions of (4.4) and (4.5), we get that

\[ \sigma_{NC}^2(\delta_n - g_n) = R_X(n) \ast (\delta_n - g_n) \ast (\delta_n - g_{-n}). \]

Then taking the DTFT of this equation we get that

\[ \sigma_{NC}^2(1 - G(\omega)) = S_X(\omega)(1 - G(\omega))^2, \tag{4.6} \]

where \( G(\omega) \) is the DTFT of \( g_n \), and \( S_X(\omega) \) is the power spectrum of \( X_n \). From equations (4.4) and (4.6), we can compute the power spectral density of both the homogeneous GMRF process and its associated non-causal prediction errors,

\[ S_X(\omega) = \frac{\sigma_{NC}^2}{1 - G(\omega)}, \tag{4.7} \]

\[ S_\xi(\omega) = \sigma_{NC}^2(1 - G(\omega)), \tag{4.8} \]

and by taking the inverse DTFT of equations (4.7), we can derive an expression for the autocorrelation function of the random process \( X_n \).

\[ R_X(n) \ast (\delta_n - g_n) = \sigma_{NC}^2\delta_n. \tag{4.9} \]
4.4 2-D Gaussian Markov Random Fields (GMRF)

In fact, all of the derivations of this section are easily generalized to regular grids in 2 or more dimensions. To do this, each lattice point is represented by $s = (s_1, s_2)$, where $s$ is a vector index with each coordinate taking on values in the range 1 to $N$. We denote the set of all lattice points as $S = \{1, \cdots, N\}^2$.

To generalize the vector-matrix relationships, we can order the pixels of the vector $X$ in raster order so that

$$X = [X_{1,1}, \cdots, X_{1,N}, X_{2,1}, \cdots, X_{2,N}, \cdots, X_{N,1}, \cdots, X_{N,N}]^t,$$

and $E$ is ordered similarly. If the GMRF is homogeneous, then the 2-D noncausal prediction error is again given by

$$E_s = X_s - \sum_{r \in \partial s} g_{s-r} X_r,$$

where $\partial s$ is a set of neighbors in 2-D. Typically, this set is given by

$$\partial s = \{r = (r_1, r_2) : r \neq s \text{ and } |r_1 - s_1| \leq P \text{ and } |r_2 - s_2| \leq P\} ,$$

where $P$ defines a $(2P+1) \times (2P+1)$ window about the point being predicted. Figure 4.2 illustrates the structure of this 2-D prediction window for an order $P$ GMRF.

Again, the prediction errors can be expressed in matrix form as

$$E = (I - G)X ,$$

where $G$ is the 2-D prediction matrix. As in the case of the 2-D AR model, the matrix $G$ is Toeplitz block Toeplitz.

The 2-D stationary GMRF also has properties quite similar to the 1-D case. In fact, the results are formally identical, only with 1-D convolution being replaced by 2-D convolution and the 1-D DTFT being replaced by the 2-D DSFT. More specifically,

$$R_E(s) = \sigma_{NC}^2 (\delta_s - g_s)$$

$$R_X(s) * (\delta_s - g_s) = \sigma_{NC}^2 \delta_s ,$$

where $*$ denotes 2-D convolution of 2-D functions. From this we can express the 2-D power spectral density of both the homogeneous GMRF process and
1-D GMRF order $P = 2$ 
2-D GMRF order $P = 2$

Figure 4.2: Structure of 1-D and 2-D GMRF prediction window for an order $P = 2$
model. The pixel denoted by the symbol $\otimes$ is predicted using both the past and
future values denoted by the symbol $\bullet$. For the 1-D case, $2P = 4$ values are used as
input to the predictor, and for the 2-D case, $(2P + 1)^2 - 1 = 24$ values are used by
the predictor.

its associated prediction errors as

\[
S_X(\mu, \nu) = \frac{\sigma^2_{NC}}{1 - G(\mu, \nu)} \quad (4.12)
\]

\[
S_\varepsilon(\mu, \nu) = \sigma^2_{NC} (1 - G(\mu, \nu)) \quad , \quad (4.13)
\]

where $G(\mu, \nu)$ is the DSFT of $g_s$.

4.5 Relation Between GMRF and Gaussian AR Models

An obvious question that arises at this point is which model is more general,
the AR or GMRF? In other words, is an AR model a GMRF, or vice versa? In
order to answer this question, we can relate the autocovariance functions and
power spectra for stationary AR and GMRF models. Table 4.1 summarizes
the important relationships from the previous chapter.

We know that if the autocovariance of the stationary AR and GMRF
processes are the same, then they must have the same distribution. Equa-
tions (3.9) and (4.9) give equations that specify the 1-D autocovariance for
the AR and GMRF cases, respectively. So if we assume that $R_X(n)$ is the
same in both expression, then we can combine these two equations to derive
the following relationship.

\[
R_X(n) * (\delta_n - g_n) \sigma^2_C = R_X(n) * (\delta_n - h_n) * (\delta_n - h_{-n}) \sigma^2_{NC}
\]

From this, we see that the equality will be satisfied if the following relationship
holds between the parameters of the AR and GMRF models.

\[ \sigma_C^2(\delta_n - g_n) = \sigma_{NC}^2(\delta_n - h_n) * (\delta_n - h_{-n}) \]  

(4.14)

So if we are given a specific AR process, then we can compute the parameters of an equivalent GMRF process using equation (4.14) above. In particular, if we evaluate the equation for \( n = 0 \), we get the following general relationship between the causal and non-causal prediction error.

\[ \sigma_{NC}^2 = \frac{\sigma_C^2}{1 + \sum_{n=1}^{P} h_n^2} \]  

(4.15)

Using this relationship, we have that

\[ g_n = \delta_n - \frac{(\delta_n - h_n) * (\delta_n - h_{-n})}{1 + \sum_{n=1}^{P} h_n^2} . \]  

(4.16)

So if we select an order \( P \) AR model, then the resulting GMRF prediction coefficients of equation (4.16) are given by the time autocorrelation of the function \( \delta_n - h_n \)

\(^2\)The autocorrelation of a function \( f_n \) is defined as \( f_n * f_{-n} \), i.e. the convolution with its time reverse.
GMRF. You should be able to convince yourself of this by working out the simple case when \( P = 2 \).

In order to find the parameters of an AR model from the parameters of a GMRF, it is necessary to find a causal predictor, \( h_n \), so that (4.14) holds in either 1-D or 2-D. In the 1-D case, this is a classic problem that has been solved for the case of Wiener filtering. Because \( g_n \) is a symmetric function, it is always possible to factor its rational Z-transform into a product of causal and anti-causal parts. However, this result cannot be generalized to 2-D because polynomials in more than one dimension cannot, in general, be factored. This leads to the following result.

**Property 4.2.** Equivalence of AR and GMRF models in 1-D - A stationary discrete time zero-mean Gaussian random process is a 1-D order \( P \) AR model if and only if it is 1-D order \( P \) GMRF.

Surprisingly, this equivalence relationship does not hold in 2-D. First, Figure 4.3 shows how a 2-D AR model of order \( P \) produces a 2-D GMRF of order \( 2P \). This is because the resulting 2-D convolution of the prediction filter produces an oblong function which is \( 4P + 1 \) wide and \( 2P + 1 \) high. However, the converse relationship simply no longer holds. That is to say that 2-D GMRFs simply are not (in general) 2-D AR processes.

**Property 4.3.** In 2-D, an AR model of order \( P \) is a GMRF of order \( 2P \) - In 2-D, a stationary discrete time zero-mean order \( P \) AR model is also an order \( 2P \) GMRF.

Figure 4.4 summarizes the situation with a Venn diagram that illustrates the relationship between AR processes and GMRFs in 1 and 2-D. So from this we see that GMRFs are more general than AR models in 2-D, but not in 1-D. This explains why they are called Gaussian Markov random fields! They grew out of a theory which was motivated by needs in 2 or more dimensions because GMRFs are more general only in 2 or more dimensions. In practice, this means that 2-D GMRFs can be used to model a broader class of distributions, which is a major justification for their use.

**Example 4.1.** Consider a zero-mean stationary AR process with order \( P = 1 \), and prediction filter

\[
h_n = \rho \delta_{n-1},
\]
Non-Causal Gaussian Models

\[
\begin{align*}
\bullet \bullet \otimes & \Rightarrow \text{1-D Autocorrelation} \Rightarrow \bullet \bullet \otimes \bullet \bullet \\
\text{1-D AR order } P = 2 & \quad \text{1-D GMRF order } P = 2 \\
\bullet \bullet \bullet \bullet \bullet & \Rightarrow \text{2-D Autocorrelation} \Rightarrow \bullet \bullet \bullet \bullet \bullet \otimes \bullet \bullet \bullet \bullet \bullet \\
\text{2-D AR order } P = 2 & \quad \text{Requires 2-D GMRF of order } 2P = 4
\end{align*}
\]

Figure 4.3: Relationship between an AR model and the corresponding GMRF in both 1-D and 2-D. In both 1-D and 2-D, the GMRF prediction window is produced by the autocorrelation of the AR filter. Notice that in 1-D, an order \(P\) AR model produces an order \(P\) GMRF. However, in 2-D an order \(P\) AR model produces a GMRF which requires an order of \(2P\). In each case, the pixels denoted by the symbol \(\otimes\) are predicted using the values denoted by the symbol \(\bullet\).

where \(|\rho| < 1\), and prediction variance \(\sigma^2_C\).

From this AR model, we would like to calculate the parameters of an equivalent GMRF. The non-causal prediction variance is given by

\[
\sigma^2_{NC} = \frac{\sigma^2_C}{1 + \rho^2}.
\]

Notice, that this non-causal prediction variance is always smaller than the causal prediction variance. The corresponding non-causal prediction filter is given by

\[
g_n = \delta_n - \frac{(\delta_n - h_n) \ast (\delta_n - h_{-n})}{1 + \rho^2} = \frac{\rho}{1 + \rho^2} (\delta_{n-1} + \delta_{n+1}).
\]

From this we can also calculate the power spectrum for both the AR and
Figure 4.4: Venn diagram illustrating the relationship between AR and GMRF models in 1 and 2-D among the set of all Gaussian random processes. Notice that in 1-D, AR and GMRF models are equivalent, but in 2-D they are not.

GMRF processes as

\[ S_X(\omega) = \frac{\sigma_C^2}{|1 - H(\omega)|^2} = \frac{\sigma_C^2}{1 - \rho e^{-j\omega}} = \frac{\sigma_C^2}{1 + \rho^2} \frac{1}{1 - \frac{2\rho}{1 + \rho^2} \cos(\omega)} , \]

or equivalently using the GMRF power spectrum

\[ S_X(\omega) = \frac{\sigma_{NC}^2}{1 - G(\omega)} = \frac{\sigma_{NC}^2}{1 - g_1 e^{-j\omega} - g_1 e^{j\omega}} = \frac{\sigma_{NC}^2}{1 - 2g_1 \cos(\omega)} = \frac{\sigma_C^2}{(1 + \rho^2) \left(1 - \frac{2\rho}{1 + \rho^2} \cos(\omega) \right)} , \]

This verifies that the two models result in the same power spectral density.

4.6 GMRF Models on General Lattices

Now that we have seen the GMRF in 1 and 2-D, we can take a slightly more abstract approach and develop a general formulation of the GMRF which is
applicable to observations indexed on any lattice. To do this, we consider a random process, $X_s$, indexed on a finite set of lattice points $s \in S$.

The neighbors of a pixel, $s$, are again denoted by $\partial s$, but we have made no specific assumptions regarding the structure of $S$, or the neighbors of a pixel, $\partial s$. In fact, our specification of neighbors must only meet the two constraints stated in the following definition.

**Definition 6. Neighborhood System**

For each $s \in S$, let $\partial s \subset S$. Then we say that $\partial s$ is a neighborhood system if it meets two conditions. First, for all $s \in S$, $s \notin \partial s$. Second, for all $s, r \in S$, if $s \in \partial r$, then $r \in \partial s$.

So for $\partial s$ to be a legitimate neighborhood system, $s$ can not be a neighbor of itself, and if $s$ is a neighbor of $r$, then $r$ must be a neighbor of $s$. Cliques are another very important concept that are very closely related to neighborhoods.

**Definition 7. Pair-Wise Clique**

An unordered pair of pixels \{s, r\} with $s, r \in S$ is said to be a pair-wise clique if $s \in \partial r$. We denote the set of all pair-wise cliques as

$$
\mathcal{P} = \{ \{s, r\} : s \in \partial r \text{ such that } s, r \in S \} .
$$

Notice that by convention, the pixel pair \{s, r\} is unordered, so each unique pair only appears once in the set $\mathcal{P}$. Using this concept of neighbors, we may give a formal definition for a Gaussian Markov random field (GMRF).

**Definition 8. Gaussian Markov random field (GMRF)**

Let $X_s$ be a jointly Gaussian random process indexed on $s \in S$. Then we say that $X$ is a Gaussian Markov random field (GMRF) with neighborhood system $\partial s$, if for all $s \in S$

$$
\mathbb{E}[X_s | X_r \text{ for } r \neq s] = \mathbb{E}[X_s | X_r \text{ for } r \in \partial s] .
$$

So a GMRF is a Gaussian random process $X_s$ such that the non-causal predictor is only dependent on neighboring values, $X_r$ for $r \in \partial s$. Using this new and somewhat more general formulation, we can restate the results of Section 4.2 that relate the non-causal prediction parameters of a zero-mean GMRF to the parameters of its density function. So the equations of non-
causal prediction become
\[ \mathcal{E}_s = X_s - \sum_{r \in \partial s} g_{s,r} X_r \]
\[ \sigma^2_s = \mathbb{E} [\mathcal{E}_s^2 \mid X_r \text{ for } r \in \partial s] \]
or in vector-matrix form this is written as
\[ \mathcal{E} = (I - G)X \]
\[ \Gamma = \text{diag} (\mathbb{E}[\mathcal{E}\mathcal{E}^t]) . \]

With this notation, we can define the following general property of zero-mean GMRFs.

**Property 4.4. Density of a zero-mean GMRF from non-causal prediction parameters** - The density function of a zero-mean Gaussian random process, \( X_s \) for \( s \in S \), with non-causal prediction matrix \( G \) and positive-definite non-causal prediction variance matrix, \( \Gamma \), is given by
\[
p(x) = \frac{1}{(2\pi)^{N/2}|B|^{1/2}} \exp \left\{ -\frac{1}{2}x^tBx \right\},
\]
where the inverse covariance, \( B \), is given by
\[
B = \Gamma^{-1}(I - G),
\]
or equivalently, the non-causal prediction parameters are given by
\[
\Gamma = \text{diag}(B)^{-1} \quad \quad G = I - \Gamma B.
\]

These relationships can also be written more explicitly as
\[
B_{s,r} = \frac{1}{\sigma^2_s} (\delta_{s-r} - g_{s,r})
\]
\[
\sigma^2_s = \frac{1}{B_{s,s}}
\]
\[
g_{s,r} = \delta_{s-r} - \sigma^2_s B_{s,r}.
\]
4.7 Chapter Problems

1. Property 3.3 applies to causal prediction errors for zero-mean Gaussian random processes. Is the same property true for non-causal prediction errors in which $\hat{X}_i = \mathbb{E}[X_i|X_k \neq i]$? If not, what step in the proof of Property 3.3 no longer holds for non-causal prediction?

2. Let $X_n$ be a 1-D zero-mean stationary Gaussian AR process with MMSE causal prediction filter given by $h_n = \rho \delta_{n-1}$ and causal prediction variance $\sigma_c^2$. Calculate $(\sigma_{NC}^2, g_n)$ the noncausal prediction variance and the noncausal prediction filter for the equivalent GMRF.

3. Let $\{X_n\}_{n=1}^5$ be a zero-mean 1-D order 2 GMRF.
   a) A friend tells you that the non-causal prediction variance for $X_n$ is $\sigma_n^2 = n$ and non-causal prediction filter is $g_n = \frac{1}{4}(\delta_{n-1} + \delta_{n+1})$. Is this possible? If so, why? If not, why not?
   b) If you know that $B_{n,m} = \sqrt{nm} \left( \delta_{n-m} - \frac{1}{4} \left( \delta_{n-m-1} + \delta_{n-m+1} \right) \right)$, then calculate the non-causal prediction variance $\sigma_n^2$ and the non-causal prediction filter $g_{m,n}$ for $X$.

4. Let $X_n$ be a zero-mean stationary GMRF with prediction filter $g_n$ and prediction variance $\sigma_{NC}^2$. Can $g_n$ be any symmetric function? If not, what properties must $g_n$ have?

5. Let $X_n$ be a zero-mean stationary 1-D Gaussian MRF with noncausal predictor $g_n$. Prove that $\sum_n g_n < 1$.

6. Let $G(\omega)$ be the DTFT of the non-causal prediction filter, $g_n$, for a zero-mean stationary GMRF. Prove that $G(\omega)$ is real valued, with $G(\omega) \leq 1$.

7. Let $X_{m,n}$ be a 2-D zero-mean wide-sense stationary random process with autocovariance function $R(k,l) = \mathbb{E}[X_{m,n}X_{m+k,n+l}]$.
   a) Show that $\forall k, l, R(k, l) = R(-k, -l)$
   b) Give an example of a wide sense stationary random process for which it is false that $\forall k, l, R(k, l) = R(-k, l)$. (Hint: This is equivalent to $\exists k, l, R(k, l) \neq R(-k, l)$.)
8. Let $X_s$ be a zero-mean 2-D Gaussian AR process indexed by $s = (s_1, s_2)$, and let the MMSE causal prediction filter be given by

$$h_s = \rho \delta_{s_1-1,s_2} + \rho \delta_{s_1,s_2-1}$$

and the causal prediction variance be given by $\sigma^2_C$. Determine $(\sigma^2_{NC}, g_s)$ the noncausal prediction variance and the noncausal prediction filter.

9. Let $X_s$ be a zero-mean GMRF on a finite general lattice $s \in S$. Let $X$ be a vector of dimension $N = |S|$ containing all the elements of $X_s$ in some fixed order, and denote the inverse covariance of $X$ as

$$B = (\mathbb{E}[XX^t])^{-1}.$$ 

a) Write an expression for $p(x)$, the PDF of $X$ in terms of $B$.

b) If $\partial s$ denotes the neighborhood system of the MRF, then show that if $r \notin \partial s$ and $r \neq s$, then $B_{r,s} = B_{s,r} = 0$.

c) Show that we can define a valid (but possibly different) neighborhood system for this GMRF as

$$\partial s = \{ r \in S : B_{r,s} \neq 0 \text{ and } r \neq s \}.$$ 

10. Let $X_s$ be a zero-mean GMRF on a finite general lattice $s \in S$ with neighborhood system $\partial s$. Let $X$ be a vector of dimension $N = |S|$ containing all the elements of $X_s$ in some fixed order. Furthermore, let $G$ be the $N \times N$ non-causal prediction matrix and let $\Gamma$ be the diagonal matrix of non-causal prediction variances.

a) Determine the inverse covariance matrix, $B$, in terms of $G$ and $\Gamma$.

b) Give conditions on $G$ and $\Gamma$ that ensure that they correspond to a valid GMRF with neighborhood system $\partial s$.

11. Let $X_s^{(k)}$ be a zero-mean GMRF on a finite general lattice $s \in S$ with neighborhood system $\partial s$. Assume the $X_{\partial s}$ is a vector of fixed dimension $P$ for all $s \in S$. Furthermore, assume that the GMRF is homogeneous so that

$$\hat{X}_s \triangleq \mathbb{E}[X_s|X_{\partial s}] = X_{\partial s}^t h$$

$$\sigma^2 \triangleq \mathbb{E}[(X_s - \hat{X}_s)^2|X_{\partial s}]$$
where $h$ is a vector of prediction filter coefficients and $\sigma^2$ is the non-causal prediction variance.

a) Determine an expression for the ML estimate of the model parameters $(h, \sigma^2)$. Is the expression in closed form? If not, what are the differences between the GMRF and the Gaussian AR model that make this ML estimate more difficult to calculate in closed form?

b) Is it true that
\[
p(x) = \prod_{s \in S} p_s(x_s | x_{\partial s})
\]
where $p_s(x_s | x_{\partial s})$ is the conditional distribution of $X_s$ given $X_{\partial s}$? How does this differ from the case of an AR model?

c) Define the log pseudo-likelihood as
\[
PL(h, \sigma^2) \triangleq \sum_{s \in S} \log p_s(x_s | x_{\partial s}, h, \sigma^2),
\]
and define the maximum pseudo-likelihood estimate of $(h, \sigma^2)$ as
\[
(\hat{h}, \hat{\sigma}^2) = \arg \max_{(h, \sigma^2)} PL(h, \sigma^2).
\]

Derive a closed form expression for the maximum pseudo-likelihood estimate in this case.
Chapter 5

MAP Estimation with Gaussian Priors

Model-based signal processing is not just about accurately modeling data. It is about using these models to extract desired information from the available data. In this chapter, we will introduce the MAP estimator as a general approach to achieving this goal. In order to illustrate the concepts, we will consider the specific application of the restoration of a blurred and noisy image. However, the framework we introduce is quite general, and can serve as the bases for the solution of any linear or non-linear inverse problem.

The MAP estimator depends on the selection of a forward and prior model. In our example, the forward model characterizes the blurring and noise processes that corrupt the measured image. The prior model provides a framework to incorporate available information about the likely behavior of the image to be restored.

We will use the GMRF of the previous chapter as a prior model since it is tractable and provides valuable insight. However, we will also see that the simple GMRF prior model has a number of practical shortcomings that limit its value in real applications. Perhaps its most important shortcoming is that the GMRF does not properly model the discontinuous edges that occur in real image data, so the associated restorations tend to be overly smooth. Luckily, this deficiency can be addressed by the non-Gaussian MRFs introduced in the next chapter.
5.1 MAP Image Restoration

Image and signal processing problems often have a common structure. Typically, there is some type of observed data, $Y$, from which one must try to determine some unknown image or signal $X$. Often, $Y$ might be the data from one or more cameras or other sensors, and $X$ might be the “true” state of the world that is of interest. For example, in medical applications, $X$ may represent noiseless and undistorted measurements of 3-D tissue properties, and $Y$ might be tomographic measurements of X-ray projections or magnetic resonance signals.

In these problems, Bayes’ rule can be used to reform the expression for the MAP estimate of equation (2.11).

$$
\hat{x}_{MAP} = \arg \max_{x \in \Omega} \{ \log p_{x|y}(x|y) \} = \arg \max_{x \in \Omega} \left\{ \log \left( \frac{p_{y|x}(y|x)p_x(x)}{p_y(y)} \right) \right\} = \arg \max_{x \in \Omega} \left\{ \log p_{y|x}(y|x) + \log p_x(x) - \log p_y(y) \right\}
$$

Since the term $\log p_y(y)$ does not depend on $x$, this term can be drop from the optimization resulting in the canonical expression for the MAP estimate.

$$
\hat{x}_{MAP} = \arg \min_{x \in \Omega} \left\{ -\log p_{y|x}(y|x) - \log p_x(x) \right\}
$$

(5.1)

In this expression, we will refer to $p_{y|x}(y|x)$ as the forward model since it quantifies how the observed data $Y$ depends on the unknown $X$. Importantly, the forward model describes how changes in $X$ can effect both the mean and the statistical distribution of observations $Y$. We refer to $p_x(x)$ in the second term as the prior model. The prior model quantifies what we know about the likely properties of the solution even before we have made any measurements. So for example, a medical image should not look like random noise. With very high probability, it should have local smoothness properties and structure that is consistent with known medical image cross-sections.

From the form of equation (5.1), we can see that the MAP estimate represents a balance between the minimization of a forward model term, $-\log p_{y|x}(y|x)$, and a prior model term, $-\log p_x(x)$. Intuitively, the MAP
5.1 MAP Image Restoration

estimate seeks to find the solution that both fits the data accurately, but also is consistent with our prior knowledge of how the solution should behave.

The MAP estimator is very powerful because it can serve as a framework for solving any problem that requires the estimation of an unknown quantity $X$ from observations $Y$. Problems of this form are known as inverse problems because they require that the effects of distorting transformations and noise be reversed. An enormous range of applications in physics, biology, chemistry, and computer science can be viewed as inverse problems; so a general theoretical framework for solving inverse problems is of great value.

In order to make the techniques of MAP estimation and inverse problems more clear, we will start with a relatively simple, but still important, problem of restoration of an image or signal $X$ from noisy and blurred data $Y$. To do this, let $X \in \mathbb{R}^N$ be an $N$ pixel image that we would like to measure, and let $Y \in \mathbb{R}^N$ be the noisy measurements given by

$$Y = AX + W$$

where $A$ is a nonsingular $N \times N$ matrix, and $W$ is a vector of i.i.d. Gaussian noise with distribution $N(0, \sigma^2I)$.

This type of inverse problem occurs commonly in imaging applications where $Y$ consists of noisy and blurred scanner measurements and $X$ is the restored image that we would like to recover. For example, the matrix $A$ may represent the blurring operation of a linear space-invariant filter. In this case, the elements of $Y$ and $X$ are related by

$$Y_s = \sum_{r \in S} A_{s-r}X_r + W_s,$$

and if the pixels are organized in raster order, then $A$ is a Toeplitz-block-Toeplitz matrix. In fact, with a little generalization, the model of (5.2) can be used to represent important problems in image reconstruction that occur in applications such as tomographic reconstruction or even astronomical imaging.

Using the model of (5.2), the conditional distribution of $Y$ given $X$ is

$$p_{y|x}(y|x) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp \left\{ -\frac{1}{2\sigma^2} ||y - Ax||^2 \right\}.$$  

(5.3)
One approach to recovering \( X \) from \( Y \) is to use the ML estimate of \( X \) given in Section 2.3:

\[
\hat{x}_{ML} = \arg \max_{x \in \mathbb{R}^N} \log p_{y|x}(y|x)
\]

\[
= \arg \max_{x \in \mathbb{R}^N} \left\{ -\frac{1}{2\sigma^2}||y - Ax||^2 - \frac{N}{2} \log(2\pi\sigma^2) \right\}
\]

\[
= \arg \min_{x \in \mathbb{R}^N} ||y - Ax||^2
\]

\[
= A^{-1}y
\]

However, this ML solution has a number of problems associated with it. First, if the matrix \( A \) is nearly singular, then inversion of \( A \) may be unstable. In order to see this, consider the singular value decomposition (SVD) of \( A \) given by

\[
A = U\Sigma V^t
\]

where \( U \) and \( V \) are orthonormal \( N \times N \) matrices and \( \Sigma = \text{diag}\{\sigma_1, \ldots, \sigma_N\} \) is a diagonal matrix of singular values arranged in monotone decreasing order.

Using this decomposition, the ML solution can be expressed as

\[
\hat{X}_{ML} = A^{-1}Y
\]

\[
= X + A^{-1}W
\]

\[
= X + V\Sigma^{-1}U^tW
\]

\[
= X + V\Sigma^{-1}\tilde{W}
\]

where \( \tilde{W} \sim N(0, \sigma^2I) \). So if the the singular values are much smaller than the noise variance (i.e., \( \sigma_N \ll \sigma \)), then the inversion process will amplify the noise in \( W \) and render the ML estimate unusable. In fact, this can easily happen if \( A \) represents a blurring filter with a transfer function that is nearly zero or small at certain frequencies. In this case, the matrix inversion is equivalent to amplification of the suppressed frequencies. Of course, this amplification increases both the signal and the noise in \( Y \), so it will produce a very noisy result. Moreover, whenever the ratio of the largest to smallest singular values is large, inversion of \( A \) will be problematic. For this reason, the ML estimate tends to be unstable whenever the condition number of the matrix \( A \) defined as

\[
\kappa(A) = \left| \frac{\sigma_1}{\sigma_N} \right|
\]
is very large. In this case, the **dynamic range** of the inverse problem can become large leading to a wide array of numerical instabilities.

However, even if $A = I$ (the identity matrix) with a condition number of 1, the ML estimate is unsatisfying because it implies that the best restoration of a noisy image is to simply accept the measured noise with no further processing. Intuitively, we might expect that some type of processing might be able to reduce the noise without excessively degrading the image detail.

An alternative approach to the ML estimate is to use the Bayesian estimator from Section 2.3.2. This approach has the advantage that one can incorporate knowledge about the probable behavior of $X$ through the selection of a prior distribution. For this example we use a zero mean homogeneous GMRF prior model for $X$, with non-causal prediction matrix $G_{i,j} = g_{i-j}$ and non-causal prediction variance $\sigma^2_x$. This results in a density function for $X$ of the form

$$p_x(x) = \frac{1}{(2\pi)^{N/2}|B|^{1/2}} \exp \left\{ -\frac{1}{2} x^t B x \right\}$$  \hspace{1cm} (5.5)

where $B_{i,j} = \frac{1}{\sigma^2_x} (\delta_{i-j} - g_{i-j})$.

Using this assumption, we can compute the posterior distribution $p_{x|y}(x|y)$. From Bayes rule, we know that

$$\log p_{x|y}(x|y) = \log p_{y|x}(y|x) + \log p_x(x) - \log p_y(y) .$$

However, this expression is difficult to evaluate directly because calculation of the term $p_y(y)$ requires the evaluation of a difficult integral. Therefore, we will take a different tack, and simply evaluate $\log p_{x|y}(x|y)$ as a function of $x$, ignoring any dependence on $y$. This means that our solution will be correct within an unknown additive constant, but we can compute that constant if needed because we know that $p_{x|y}(x|y)$ must integrate to 1. So using expressions (5.3) and (5.5), we have that

$$\log p_{x|y}(x|y) = -\frac{1}{2\sigma^2} ||y - Ax||^2 - \frac{1}{2} x^t B x + c(y)$$ \hspace{1cm} (5.6)

where $c(y)$ is some as yet unknown function of $y$. Since (5.6) is a quadratic function of $x$, we can complete-the-square and express this function in the standard quadratic form

$$\log p_{x|y}(x|y) = -\frac{1}{2} (x - \mu(y))^t R^{-1} (x - \mu(y)) + c'(y)$$ \hspace{1cm} (5.7)
where \( \mu(y) \) and \( R \) are given by

\[
\begin{align*}
\mu(y) &= (A^t A + \sigma^2 B)^{-1} A^t y \\
R &= \left( \frac{1}{\sigma^2} A^t A + B \right)^{-1},
\end{align*}
\]

and again \( c'(y) \) is only a function of \( y \). From the form of (5.7), we know that \( p_{x|y}(x|y) \) must be conditionally Gaussian, with a conditional mean of \( \mu(y) \) and conditional variance of \( R \). Therefore, the conditional density must have the form

\[
p_{x|y}(x|y) = \frac{1}{(2\pi)^{N/2}|R|^{-1/2}} \exp \left\{ -\frac{1}{2} (x - \mu(y))^t R^{-1} (x - \mu(y)) \right\}. \tag{5.8}
\]

Using the posterior distribution of (5.8), we can compute some typical Bayesian estimators. Notice that the MMSE estimator is given by the conditional mean,

\[
\hat{x}_{\text{MMSE}} = E[X|Y = y] = \mu(y),
\]

and the MAP estimate is given by the conditional mode

\[
\hat{x}_{\text{MAP}} = \arg \max_{x \in \mathbb{R}^N} \log p_{X|Y}(x|y) \\
= \arg \min_{x \in \mathbb{R}^N} \left\{ \frac{1}{2} (x - \mu(y))^t R^{-1} (x - \mu(y)) \right\} \\
= \mu(y).
\]

Notice that for this case, the MMSE and MAP estimates are the same. In fact, this is always the case when all the random quantities in an estimation problem are jointly Gaussian. However, we will find that the MMSE and MAP are generally not the same when the distributions are non-Gaussian. This will be important later since we will see that non-Gaussian distributions are better models for image features such as edges.

### 5.2 Computing the MAP Estimate

So it seems that our problem is solved with the ubiquitous estimate

\[
\hat{x} = (A^t A + \sigma^2 B)^{-1} A^t y. \tag{5.9}
\]
However, while the solution of (5.9) is mathematically appealing it is not very practical for most applications because the dimension of the matrix to be inverted is enormous. For example, if \( N \) equals 1 million pixels, then the matrix to be inverted has \( 10^{12} \) elements. If the matrix is dense and each element is stored with a 64 bit float, then it will require approximately 8 terrabytes of storage! So, clearly we will need another approach.

In order to make the computation of the MAP estimate more tractable, we must go back to the first principles of its derivation. The MAP estimate of \( X \) given \( Y \) is given by the optimization of equation (5.1). So substituting the expressions from (5.3) and (5.5) into (5.1), yields the equation

\[
\hat{x}_{MAP} = \arg \min_{x \in \mathbb{R}^N} \left\{ \frac{1}{2\sigma^2} ||y - Ax||^2 + \frac{1}{2} x^t B x \right\}. \tag{5.10}
\]

From equation (5.10), we see that, in this case, MAP estimation is equivalent to minimization of a quadratic function of the form

\[
f(x) = \frac{1}{2\sigma^2} ||y - Ax||^2 + \frac{1}{2} x^t B x. \tag{5.11}
\]

The key insight is that while the direct inversion of (5.9) provides an exact mathematical solution, it requires an enormous amount of computation to evaluate. Alternatively, numerical optimization methods can provide an approximate solution to the minimization of (5.10) that can be computed much more efficiently.

In the following sections, we present four typical methods for solving this optimization problem. While many other methods exist, these methods are selected to provide canonical examples of the strengths and weaknesses of various optimization approaches.

### 5.3 Gradient Descent Optimization

A natural approach to optimization of a continuously differentiable function is to start at an initial value \( x^{(0)} \), and then incrementally move in the opposite direction of the gradient. Intuitively, the gradient is the direction of the function’s most rapid increase, so one would expect moving in the opposite direction to be an effective method for minimizing the function. This approach to optimization is generally referred to as **gradient descent** optimization.
The general form of gradient descent iterations is given by
\[ x^{(k+1)} = x^{(k)} - \beta \nabla f(x^{(k)}) \]
where \( x^{(k)} \) is the current image estimate, \( x^{(k+1)} \) is the updated image, and \( \beta \) is a positive step size that determines how far one moves along the negative gradient direction. For the image restoration problem of (5.10), the gradient of \( f(x) \) is given by
\[ \nabla f(x) = -\frac{1}{\sigma^2} A^t(y - Ax) + Bx . \] (5.12)
So for this case, the gradient descent update equation is given by
\[ x^{(k+1)} = x^{(k)} + \alpha [A^t(y - Ax^{(k)}) - \sigma^2 Bx^{(k)}] \] (5.13)
where \( \alpha = \beta/\sigma^2 \) is a scaled version of the step size. Rearranging terms, we have the recursion
\[ x^{(k+1)} = (I - \alpha [A^t A + \sigma^2 B])x^{(k)} + \alpha A^t y . \]
An important special case of (5.14) occurs when \( A \) represents a linear space-invariant filter with 2-D impulse response \( a_s \). When \( A \) represents a space-invariant filter, then it has a Toeplitz-block-Toeplitz structure, and the operation \( Ax \) is equivalent to the truncated 2-D convolution \( a_s * x_s \). Similarly, multiplication by the transpose, \( A^t y \), is equivalent to 2-D convolution with the space-reversed impulse response, \( a_{-s} * y_s \). In this case, the gradient descent update equation of (5.13) has the form
\[ x_s^{(k+1)} = x_s^{(k)} + \alpha [a_{-s} * (y_s - a_s * x_s^{(k)}) - \lambda (\delta_s - g_s) * x_s^{(k)}] , \] (5.14)
where \( g_s \) is the non-causal predictor for the GMRF and \( \lambda = \sigma^2/\sigma_x^2 \) is a measure of the noise-to-signal ratio. Here you can see that each iteration of gradient descent is computed through the application of LSI filters; so as long as the filter impulse response is spatially limited, the computation is not excessive.

5.3.1 Convergence Analysis of Gradient Descent

Gradient descent is a simple algorithm, but it has a key disadvantage related to the difficulty in selection of the step-size parameter. In order to better
5.3 Gradient Descent Optimization

Figure 5.1: 3-D rendering of the example non-quadratic function from two different views. The analytical form of the function is \( \frac{1}{(x-y)^2+1} \exp(-4(x+y)^2) \).

![3-D rendering](image)

Figure 5.2: 2-D contour plots of the trajectory of gradient descent optimization using a step size of a) \( \alpha = 0.02 \); b) \( \alpha = 0.06 \); c) \( \alpha = 0.18 \). Notice that the convergence with small \( \alpha \) is slow, and the convergence with large \( \alpha \) is unstable.

![2-D contours](image)

illustrate this issue, consider the minimization of the 2-D function \( c(x, y) = -\frac{1}{(x-y)^2+1} \exp(-4(x+y)^2) \). Some intuition into the structure of this function can be gleaned from Figure 5.1 which shows a 3-D rendering of the negative of the function, \(-c(x, y)\). First notice that the function has a sharp peak but becomes relatively flat in regions further from its maximum. Also, it should be noted that maximization of \(-c(x, y)\) (using for example gradient ascent) is equivalent to minimization of \(c(x, y)\) (using for example gradient descent).

Figure 5.2 shows the corresponding convergence behavior of gradient descent for three different step sizes. Too large of a step size results in the unstable convergence shown in Figure 5.2c, whereas too small of a step size results in the slow convergence of Figure 5.2a. Unfortunately, for high dimensional optimization problems such as (5.10), we will see that there is often no good choice of \( \alpha \).
To better understand why this is true, we can analyze the convergence behavior of the gradient descent algorithm. Let us assume that the iteration of (5.14) converges to the limit \( x^{(\infty)} \), then we can define the error in the calculation of the MAP estimate to be \( \epsilon^{(k)} = x^{(k)} - x^{(\infty)} \). In this case, we know that taking the limit of (5.14) results in

\[
x^{(\infty)} = (I - \alpha[A^tA + \sigma^2B])x^{(\infty)} + \alpha A^ty .
\]  

(5.15)

Subtracting (5.14) and (5.15) results in

\[
\epsilon^{(k+1)} = (I - \alpha[A^tA + \sigma^2B]) \epsilon^{(k)} .
\]  

(5.16)

We can further simplify the structure of (5.16) by using eigenvector analysis of the matrix

\[ H \overset{\Delta}{=} A^tA + \sigma^2B . \]

Since \( H \) is symmetric, we know it can be diagonalized as \( H = E\Lambda E^t \) where columns of \( E \) are the orthonormal eigenvectors of \( H \), and \( \Lambda = \text{diag}\{h_1, \cdots, h_N\} \) is a diagonal matrix containing the corresponding \( N \) eigenvalues of \( H \). Notice that the eigenvalues, \( h_i \), are all strictly positive because \( H \) is formed by the sum of two positive-definite matrices. If we define the transformed error as \( \tilde{\epsilon}^{(k)} = E^t\epsilon^{(k)} \), then we may derive the following simple relationship for the decay of the error in the gradient descent method.

\[
\tilde{\epsilon}^{(k)} = (I - \alpha \Lambda)^k \tilde{\epsilon}^{(0)} .
\]  

(5.17)

Since \( \Lambda \) is diagonal, we can see that the \( i^{th} \) component of the error then decays as

\[
\tilde{\epsilon}_i^{(k)} = (1 - \alpha h_i)^k \tilde{\epsilon}_i^{(0)} .
\]  

(5.18)

Equation (5.18) provides great insight into the convergence of gradient descent to the MAP estimate. First, in order for convergence to be stable, we must have that \(|1 - \alpha h_i| < 1\) for all \( i \). Second, in order for convergence to be rapid, we would like that \(1 - \alpha h_i \approx 0\) for all \( i \). As we will see, it is easy to make convergence fast for one \( i \), but it is generally not possible to make it fast for all \( i \).

A typical approach is to select \( \alpha \) so that convergence is fast for the largest eigenvalue, \( h_{max} = \max_i\{h_i\} \). To do this we select \( \alpha \) so that \(1 - \alpha h_{max} = 0\), which implies that

\[
\alpha = \frac{1}{h_{max}} .
\]
5.3 Gradient Descent Optimization

Figure 5.3: Plots showing (a) the frequency response of $h(\omega)$, (b) the resulting function $1 - h(\omega)/h_{\text{max}}$ which determines the convergence rate of gradient descent for a step-size with good low frequency convergence, (c) the number of iterations of gradient descent required for 99% convergence. Notice that gradient descent has slow convergence at high frequencies. In fact, this tends to be a general property of gradient descent for deblurring problems.

However, when we do this the mode associated with the smallest eigenvalue, $h_{\text{min}} = \min_i \{h_i\}$, has very slow convergence. To see this, let $i_{\text{min}}$ be the index of the smallest eigenvalue, then the convergence of the associated mode is given by

$$\frac{\epsilon_{i_{\text{min}}}^{(k)}}{\epsilon_{i_{\text{min}}}^{(0)}} = (1 - \alpha h_{\text{min}})^k = \left(1 - \frac{h_{\text{min}}}{h_{\text{max}}} \right)^k = \left(1 - \frac{1}{\kappa(H)} \right)^k$$

where $\kappa(H) = \frac{h_{\text{max}}}{h_{\text{min}}}$ is the condition number of the matrix $H$. So if $\kappa$ is very large, then $1 - \frac{1}{\kappa(H)}$ is very close to 1, and the convergence for the mode corresponding to the smallest eigenvalue will be very slow.

In many applications, the condition number of $H$ can be very large. When this is the case, it is not possible to choose $\alpha$ to achieve fast and stable convergence for both large and small eigenvalues. In general, small eigenvalues will converge slowly; and if one tries to speed the convergence of small eigenvalues by increasing $\alpha$, then the modes with large eigenvalues can become unstable.

5.3.2 Frequency Analysis of Gradient Descent

In many practical situations, the matrix $A$ represents a linear space-invariant filter applied over a large region. In this case, the convolutions of equa-
tion (5.14) can be approximated by multiplication in the frequency domain, and it is possible to gain more insight into the convergence behavior of gradient descent. More specifically, taking the DSFT of (5.14) and rearranging terms results in the following relationship.

$$x^{(k+1)}(\omega) = (1 - \alpha \left[ |a(\omega)|^2 + \lambda (1 - g(\omega)) \right]) x^{(k)}(\omega) + \alpha a^*(\omega) y(\omega) \quad (5.19)$$

If we further define $\varepsilon^{(k)}(\omega)$ to be the DSFT of $\varepsilon^{(k)}_s = x^{(k)}_s - x^\infty_s$, then we can derived a recursion for the error in the computation of the MAP estimate.

$$\varepsilon^{(k+1)}(\omega) = (1 - \alpha \left[ |a(\omega)|^2 + \lambda (1 - g(\omega)) \right]) \varepsilon^{(k)}(\omega) \quad (5.20)$$

Using the definition that

$$h(\omega) \triangleq |a(\omega)|^2 + \lambda (1 - g(\omega)) \quad (5.21)$$

we have that

$$\varepsilon^{(k)}(\omega) = (1 - \alpha h(\omega))^k \varepsilon^{(0)}(\omega) .$$

So here we can see that the values of the frequency transform $h(\omega)$ are essentially the eigenvalues of $H$. Furthermore, we know that $h(\omega) > 0$ due to the fact that $1 - g(\omega) > 0$.

If we again define $h_{\text{max}} = \max_\omega h(\omega)$ and $h_{\text{min}} = \min_\omega h(\omega)$, then we can select $\alpha$ for fast convergence for the largest eigenvalue which results in

$$\varepsilon^{(k)}(\omega) = \left( 1 - \frac{h(\omega)}{h_{\text{max}}} \right)^k \varepsilon^{(0)}(\omega) .$$

Example 5.1. Consider a 1-D signal deconvolution problem of the form

$$Y_n = \sum_{k=-1}^{1} a_k X_{n-k} + W_n ,$$

where

$$a_n = \delta_n + \frac{1}{2} (\delta_{n-1} + \delta_{n+1})$$

is a smoothing filter, and $W_n$ is additive noise modeled as i.i.d. Gaussian random variables with $N(0, \sigma^2)$ distribution. We will use a prior model for
X of a GMRF with a non-causal prediction variance of $\sigma_x^2$ and a non-causal prediction filter of the form

$$g_n = \frac{1}{2} (\delta_{n-1} + \delta_{n+1}) .$$

Taking the DTFT of $a_n$ and $g_n$ results in the following expressions.

$$a(\omega) = 1 + \cos(\omega)$$
$$1 - g(\omega) = 1 - \cos(\omega)$$

If we assume that $\sigma^2 = 1$ and $\sigma_x^2 = 10$, then $\lambda = \sigma^2 / \sigma_x^2 = 1/10$, and we can calculate the following explicit expression for $h(\omega)$ by using equation (5.21).

$$h(\omega) \triangleq |1 + \cos(\omega)|^2 + \frac{1}{10} (1 - \cos(\omega))$$

Figure 5.3(a) shows a plot of $h(\omega)$. Notice that the function has a low pass nature because it is computed from the blurring kernel $a_n$. The maximum of $h(\omega)$ occurs at $\omega = 0$ and the minimum at $\omega = \pm \pi$, and these values correspond to $h_{\text{max}} = 4$ and $h_{\text{min}} = 1/5$. If we set the step size to $\alpha = 1/h_{\text{max}} = 1/4$, then the convergence at frequency $\omega$ is given by

$$\epsilon^{(k)}(\omega) = \left(1 - \frac{h(\omega)}{4}\right)^k \epsilon^{(0)}(\omega) .$$

Figure 5.3(b) shows a plot of the function $1 - \frac{h(\omega)}{4}$. Notice that it takes on its largest values at $\omega = \pm \pi$. This means the convergence will be slowest at the high frequencies. Figure 5.3(c) makes this clearer by plotting the number of iterations required for 99% convergence of the gradient descent algorithm as a function of frequency. Notice that at $\omega = 0$, convergence is in 1 iteration; but at a frequency of $\omega = \pi$, gradient descent takes over 90 iterations to converge.

This result of Example 5.1 is typical for the application of gradient descent to deconvolution or deblurring problems. Low spatial frequencies tend to converge rapidly, but high spatial frequencies tend to converge slowly. When solving 2-D problems, this means that the low frequency shading tends to converge quickly, but the high frequency detail, such as edges, converge slowly. This is a serious problem in imaging problems because typically edge content is particularly important in the perceived quality of an image.
5.3.3 Gradient Descent with Line Search

One partial solution to the problem of selecting a step size is to choose the value of $\alpha$ at each iteration that minimizes the cost function. In order to do this, we will need to introduce a very important concept known as line search for finding the minimum of any 1-D cost function. The three update equations for gradient descent with line search are listed below.

\begin{align*}
    d^{(k)} &= -\nabla f(x^{(k)}) \quad (5.22) \\
    \alpha^{(k)} &= \arg\min_{\alpha \geq 0} f(x^{(k)} + \alpha d^{(k)}) \quad (5.23) \\
    x^{(k+1)} &= x^{(k)} + \alpha^{(k)} d^{(k)} \quad (5.24)
\end{align*}

The first step of equation (5.22) determines the update direction, $d^{(k)}$. The second step of equation (5.23) then determines the value of the step size that minimizes the cost function along the direction of the search. This step is known as line search since it searches along a line in the direction of $d^{(k)}$. It is often the case that the line search can actually dominate the computation in an optimization strategy; so the design of effective line search strategies will be an important theme to come. The third step of equation (5.24) then updates the state to form $x^{(k+1)}$.

One approach to solving the line search problem is to root the derivative of the 1-D function. This can be done by finding a value of $\alpha$ which solves the following equation.

\begin{equation}
    0 = \frac{\partial f(x^{(k)} + \alpha d^{(k)})}{\partial \alpha} = \left[ \nabla f(x^{(k)} + \alpha d^{(k)}) \right]^t d^{(k)} \quad (5.25)
\end{equation}
In some cases, the line search required for steepest descent can be computationally intensive, but for quadratic optimization it can be computed in closed form with modest computation. Applying the gradient expression of (5.25) to the cost function of (5.11) results in the following closed form expression for the step size,

\[
\alpha^{(k)} = \frac{(y - Ax^{(k)})^t A d^{(k)} - \sigma^2 [x^{(k)}]^t B d^{(k)}}{||d^{(k)}||_H^2} ,
\]

(5.26)

where \(H = A^t A + \sigma^2 B\), and we use the notation \(||z||_W^2 = z^t W z\) for a positive-definite matrix \(W\). Notice that the expression of equation (5.26) holds for any choice of direction \(d^{(k)}\) as long as the cost function of (5.11) is used. This is important because in later sections we will introduce methods such as conjugate gradient that introduce modifications to the gradient.

However, for the image restoration problem of (5.10), the gradient direction is given by

\[
d^{(k)} = \frac{1}{\sigma^2} A^t (y - Ax^{(k)}) - B x^{(k)} ,
\]

(5.27)

and in this case, the step size has an even simpler form given by

\[
\alpha^{(k)} = \sigma^4 \frac{||d^{(k)}||_W^2}{||d^{(k)}||_H^2} .
\]

(5.28)

Figure 5.4(a) illustrates the results of gradient descent with line search for the cost function of Figure 5.1. Notice that the convergence of steepest descent cost function is guaranteed to be stable because with each iteration the cost function is monotone decreasing.

\[
f(x^{(k+1)}) \leq f(x^{(k)})
\]

If the cost function \(f(x)\) is bounded below, as is the case for our problem, then we know that \(\lim_{k \to \infty} f(x^{(k)})\) converges.

### 5.4 Iterative Coordinate Descent Optimization

As we saw in Section 5.3.2, gradient descent tends to have slow convergence at high spatial frequencies. In practice, most gradient based methods tend to
converge faster at low spatial frequencies for problems that require deblurring
or deconvolution of an image. An alternative optimization approach with
to meet the constraint of the differential equation. The Gauss-Seidel method is in many ways analogous to
faster convergence at high spatial frequencies is the **iterative coordinate
descent (ICD)** algorithm. ICD works by updating each pixel in sequence
while keeping the remaining pixels fixed. One full iteration of ICD then
consists of a sequence of single updates, were each pixel in $x$ is updated
exactly once.

The ICD algorithm is most easily specified using the assignment notation
of a programming language. Using this notation, the value of the pixel $x_s$ is updated using the rule

$$ x_s \leftarrow \arg \min_{x_s \in \mathbb{R}} f(x) $$

while the remaining pixels $x_r$ for $r \neq s$ remain unchanged. The result of an
ICD update of the pixel $s$ can be compactly expressed as $x + \alpha \varepsilon_s$ where $\varepsilon_s$
is a vector that is 1 for element $s$, and 0 otherwise. Using this notation, we have that

$$ x_s \leftarrow x_s + \arg \min_{\alpha \in \mathbb{R}} f(x + \alpha \varepsilon_s) . $$

The ICD update can then be computed by solving the equation

$$ 0 = \frac{\partial f(x + \alpha \varepsilon_s)}{\partial \alpha} = [\nabla f(x + \alpha \varepsilon_s)]^t \varepsilon_s , $$

which results in

$$ \alpha = \frac{(y - Ax)^t A_{s,s} - \sigma^2 x^t B_{s,s}}{||A_{s,s}||^2 + \sigma^2 B_{s,s}} , \tag{5.29} $$

where $A_{s,s}$ and $B_{s,s}$ denote the $s^{th}$ column of $A$ and $B$ respectively. This then
results in the ICD update equation

$$ x_s \leftarrow x_s + \frac{(y - Ax)^t A_{s,s} - \lambda (x_s - \sum_{r \in \partial s} g_{s-r} x_r)}{||A_{s,s}||^2 + \lambda} , \tag{5.30} $$

where $\lambda = \frac{\sigma_2^2}{\sigma_1^2}$ is the effective noise-to-signal ratio.

Direct implementation of the ICD algorithm using (5.30) requires the evaluation
of the term $y - Ax$ with each pixel update. Typically, this requires

---

1 The Gauss-Seidel algorithm for solving differential equations works by updating each point in sequence
to meet the constraint of the differential equation. The Gauss-Seidel method is in many ways analogous to
ICD, and the techniques for analyzing the two algorithms are closely related.
ICD Algorithm:

- Initialize \( e \leftarrow y - Ax \)
- For \( K \) iterations {
  - For each pixel \( s \in S \) {
    - \( v \leftarrow x_s \)
    - \( x_s \leftarrow x_s + \frac{e^t A_{s,s} - \lambda (x_s - \sum_{r \in \partial s} g_{s-r} x_r)}{||A_{s,s}||^2 + \lambda} \)
    - \( e \leftarrow e - A_{s,s} (x_s - v) \)
  }
}

Figure 5.5: Pseudocode for fast implementation of the ICD algorithm. The speedup depends on the use of a state variable to store the error, \( e = y - Ax \).

too much computation. Fortunately, there is a simple method for speeding the computation by keeping a state variable which stores the error term \( e = y - Ax \). The pseudocode for this faster version of ICD is shown in Figure 5.5.

As with gradient descent, an important special case of (5.30) occurs when \( A \) represents a linear space-invariant filter \( a_s \). In this case, multiplication by \( A \) is equivalent to convolution with \( a_s \); multiplication by \( A^t \) is equivalent to convolution with \( a_{-s} \); and the ICD update has the form

\[
x_s \leftarrow x_s + \frac{a_{-s} * (y_s - a_s * x_s) - \lambda (x_s - \sum_{r \in \partial s} g_{s-r} x_r)}{\sum_s a_s^2 + \lambda}.
\tag{5.31}
\]

When the point spread function is \( a_s = \delta_s \), then there is no blurring, and the update equation has a simpler form which lends insight into the process.

\[
x_s \leftarrow \frac{y_s + \lambda \sum_{r \in \partial s} g_{s-r} x_r}{1 + \lambda}
\]

Notice that in this case, the ICD update is a weighted average between the measurement, \( y_s \), and the non-causal prediction, \( \sum_{r \in \partial s} g_{s-r} x_r \). When \( \lambda \) is large and the signal-to-noise is low, then the update is weighted toward the neighboring pixel values; however, when \( \lambda \) is small and the signal-to-noise is high, then the update is weighted toward the measured data.
5.4.1 Convergence Analysis of Iterative Coordinate Descent

It is also possible to analyze the convergence of ICD, but the analysis is a bit more tricky [59]. We first define the matrix

\[ H \triangleq A^t A + \sigma^2 B. \]

Using the gradient expression of (5.12), the scaled gradient of \( f(x) \) can then be expressed as

\[ \sigma^2 \nabla f(x) = -A^t y + Hx. \]

The objective of coordinate descent is to minimize the cost as a function of the variable \( x_s \). So we enforce this condition by setting the \( s^{th} \) gradient term to 0.

\[ [\nabla f(x)]_s = [-A^t y + Hx]_s = 0 \]

This can be more explicitly written as

\[ -\sum_{r \in S} a_{r,s} y_r + \sum_{r \in S} H_{s,r} x_r = 0. \] (5.32)

Now, let us assume that the ICD updates are performed in order. When we update the pixel \( x_s \), the values of the previous pixels \( x_r \) for \( r < s \) will have already been updated, and the values of \( x_r \) for \( r > s \) have yet to be updated. This means that at the \( k^{th} \) iteration, the past pixels (i.e., \( r < s \)) take on their new value, \( x_r = x_r^{(k+1)} \), and the future pixels (i.e., \( r > s \)) still retain their old
value, \( x_r = x_r^{(k)} \). Substituting these values into (5.32) results in the following ICD update equation,

\[
- \sum_{r \in S} a_{r,s}y_r + \left[ \sum_{r<s} H_{s,r}x_r^{(k+1)} \right] + H_{s,s}x_s^{(k+1)} + \left[ \sum_{r>s} H_{s,r}x_r^{(k)} \right] = 0 ,
\]

(5.33)

where notice that \( x_s = x_s^{(k+1)} \) because the result of solving this equation will be to determine the updated value of \( x_s \).

Now since equation (5.33) holds for all values of \( r \) and \( s \), it can be expressed more simply in matrix notation. Specifically, we can partition \( H \) into its lower triangular, upper triangular, and diagonal portions. So let \( L_{r,s} = 0 \) for \( r \leq s \) be the lower triangular portion of \( H \), and let \( D_{r,s} = 0 \) for \( r \neq s \) be its diagonal portion. Since \( H \) is symmetric, we know that its upper triangular portion must be given by \( L^t \). Then we have that

\[
H = L + D + L^t .
\]

Using this notation, equation (5.33) becomes

\[
-A^ty + (L + D)x^{(k+1)} + L^tx^{(k)} = 0 .
\]

So the update equation for \( x^{(k)} \) becomes

\[
x^{(k+1)} = -(L + D)^{-1}(L^tx^{(k)} - A^ty) .
\]

If we define \( \epsilon^{(k)} = x^{(k)} - x^{(\infty)} \), then this equation becomes

\[
\epsilon^{(k+1)} = -(L + D)^{-1}L^t\epsilon^{(k)} .
\]

(5.34)

The convergence of this equation can be studied in much the same way that we analyzed the convergence of equation (5.16) for gradient descent.

In order to illustrate this, we will consider the case when \( A \) and \( B \) are Toeplitz matrices. In this case, the convergence of ICD can be analyzed in the frequency domain. Since \( A \) is Toeplitz, multiplication by \( A \) is equivalent to convolution with \( a_s \). We will also assume that \( B \) is equivalent to convolution with \( \frac{1}{\sigma^2}(\delta_s - g_s) \). So we have that multiplication by \( H \) is equivalent to convolution by \( h_s \) where

\[
h_s = a_s * a_{-s} + \lambda(\delta_s - g_s) ,
\]

(5.35)
where $\lambda = \sigma^2 / \sigma^2_x$. Next we can define the functions corresponding the causal, and memoryless parts of $h_s$.

$$l_s = \begin{cases} h_s & \text{if } s < 0 \\ 0 & \text{if } s \geq 0 \end{cases}, \quad d_s = h_0 \delta_s$$

Then $h_s = l_s + l_{-s} + d_s$. The corresponding DSFTs of these functions then become $L(\omega)$ and $D(\omega) = h_0$, respectively. Using these Fourier transforms, we can then represent the update equation for the error in the ICD convergence as

$$\epsilon^{(k+1)}(\omega) = \left( \frac{-L^*(\omega)}{L(\omega) + h_0} \right)^k \epsilon^{(0)}(\omega).$$

So from this, we can see that the rate of convergence of the ICD algorithm is determined by the magnitude of the term

$$P(\omega) = \left| \frac{L^*(\omega)}{L(\omega) + h_0} \right|. \quad (5.36)$$

If $P(\omega)$ is near zero, then convergence is rapid, but if it is near 1, then convergence is slow.

**Example 5.2.** Again consider the problem of 1-D deconvolution where our objective is to recover a sequence $X_n$ from observations $Y_n$ defined by

$$Y_n = \sum_{k=-1}^{1} a_k X_{n-k} + W_n,$$

where

$$a_n = \delta_n + \frac{1}{2} \left( \delta_{n-1} + \delta_{n+1} \right),$$

$W_n$ is i.i.d. noise distributed as $N(0, \sigma^2)$, and $X$ is a GMRF with a non-causal prediction variance of $\sigma^2_x$ and a non-causal prediction filter of the form

$$g_n = \frac{1}{2} \left( \delta_{n-1} + \delta_{n+1} \right).$$

If we assume that $\sigma^2 = 1$ and $\sigma^2_x = 10$, then we can compute the function $h_s$.
of equation (5.35) as

$$h_n = a_n * a_{-n} + \frac{\sigma^2}{\sigma^2_x} (\delta_n - g_n)$$

$$= \frac{3}{2} \delta_n + (\delta_{n-1} + \delta_{n+1}) + \frac{1}{4} (\delta_{n-2} + \delta_{n+2}) + \frac{1}{10} \delta_n - \frac{1}{20} (\delta_{n-1} + \delta_{n+1})$$

$$= \frac{8}{5} \delta_n + \frac{19}{20} (\delta_{n-1} + \delta_{n+1}) + \frac{1}{4} (\delta_{n-2} + \delta_{n+2}) .$$

So from this, we have that

$$l_s = \frac{19}{20} \delta_{n-1} + \frac{1}{4} \delta_{n-2} , \quad d_s = \frac{8}{5} \delta_n .$$

From this, we can calculate the function $P(\omega)$ from equation (5.36).

Figure 5.6(b) shows a plot of the function $P(\omega)$. Notice that its value is substantially less than 1 for all $\omega$. This means the convergence will be relatively fast for a wide range of frequencies. Figure 5.6(c) makes this clearer by plotting the number of iterations required for 99% convergence of the ICD algorithm as a function of frequency. Comparing this to Figure 5.3(c), we can see that ICD’s high frequency convergence is much more rapid than gradient descent, with gradient descent requiring over 80 iterations and ICD requiring less than 20.

This result of Example 5.2 is typical of ICD which tends to have relatively rapid convergence at high spatial frequencies for this type of problem. When solving 2-D problems, this means that the high frequency detail, such as edges, converge quickly. However, in some cases this can come at the cost of slower convergence at low spatial frequencies. This is why it can often be helpful to use an initialization for ICD that accurately approximates the low frequency components to the MAP estimate.

### 5.5 Conjugate Gradient Optimization

One of the most popular and effective methods for doing optimization is known as **conjugate gradient (CG)**. As we saw in Section 5.3.1 the convergence speed of gradient descent algorithms tends to be limited by the
condition number of the matrix $H$. The CG method attempts to mitigate this problem by enforcing the constraint that each new update is conjugate to the previous updates.

The theory of CG goes beyond this treatment, but the general form of the update equations are quite simple. The algorithm is initialized by setting $h^{(0)} = d^{(0)} = -\nabla f(x^{(0)})$ where $x^{(0)}$ is the initial value of the image $x$. Then the following iteration is run until the desired level of convergence is achieved.

$$d^{(k)} = -\nabla f(x^{(k)})$$

$$\gamma_k = \frac{(d^{(k)} - d^{(k-1)})^t d^{(k)}}{(d^{(k-1)})^t d^{(k-1)}}$$

$$h^{(k)} = d^{(k)} + \gamma_k h^{(k-1)}$$

$$\alpha^{(k)} = \arg\min_{\alpha \geq 0} f(x^{(k)} + \alpha h^{(k)})$$

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} h^{(k)}$$

It can be shown that when $f(x)$ is a quadratic function of $x$, then the update directions, $h^{(k)}$, are conjugate, which is to say that the following relationship holds,

$$(h^{(i)})^t H h^{(j)} = 0 \quad \text{for} \ i \neq j,$$

and the algorithm converges to the global optimum in no more than $N$ steps! Of course, in many cases, it would not be practical to run $N$ iterations, but it is both interesting and reassuring to know that convergence is achieved in a finite number of iterations.
5.6 Chapter Problems

1. Derive the expressions for $\mu(y)$ and $R$ used in equation (5.7).

2. Consider a MAP estimation problem where the observations are given by $Y = AX + W$ where $A \in \mathbb{R}^{M \times N}$ and $X$ and $W$ are independent jointly Gaussian random vectors with distributions given by $X \sim N(0, B^{-1})$, $W \sim N(0, \Lambda^{-1})$, where both $B \in \mathbb{R}^{M \times M}$ and $\Lambda \in \mathbb{R}^{M \times M}$ are symmetric and positive-definite matrix.
   a) Write an expression for the conditional density, $p_{y|x}(y|x)$.
   b) Write an expression for the prior density, $p_x(x)$.
   c) Write an expression for the MAP cost function, $f(x)$, for this problem.

3. Consider the function
   \[ f(x) = \frac{1}{2} \|y - Ax\|^2_\Lambda + \frac{1}{2} x^t B x , \]
   where $y \in \mathbb{R}^M$, $x \in \mathbb{R}^N$, and $B \in \mathbb{R}^{N \times N}$ is a symmetric positive semi-definite matrix.
   a) Calculate $\nabla f(x)$, the gradient of the cost function.
   b) Calculate $H(x)$, the Hessian matrix of the cost function given by
      \[ [H(x)]_{s,r} = \frac{\partial^2 f(x)}{\partial x_s \partial x_r} \]
   c) Calculate the global minimum of the cost function.

4. Show that the update step for gradient descent with line search is given by equations (5.26) and (5.28).

5. Show that the ICD update step is given by equation (5.29).

6. Let $X \in \mathbb{R}^N$ be an $N$ pixel image that we would like to measure, and let $Y \in \mathbb{R}^N$ be the noisy measurements given by
   \[ Y = AX + W \]
   where $A$ is an $N \times N$ nonsingular matrix, $W$ is a vector of i.i.d. Gaussian noise with $W \sim N(0, \Lambda^{-1})$. Furthermore, in a Bayesian framework,
assume that $X$ is a GMRF with noncausal prediction filter $g_s$ and noncausal prediction variance $\sigma^2$.

a) Derive an expression for the ML estimate of $X$.

b) Derive an expression for the MAP estimate of $X$ under the assumption that $X$ is a zero mean GMRF with inverse covariance matrix $B$.

c) Derive an expression for the MAP cost function $f(x)$.

7. Consider the optimization problem

$$\hat{x} = \arg\min_{x \in \mathbb{R}^N} \left\{ \| y - Ax \|^2_\Lambda + x^t B x \right\}$$

where $A$ is a nonsingular $N \times N$ matrix, $B$ is a positive-definite $N \times N$ matrix, and $\Lambda$ is a diagonal and positive-definite matrix.

a) Derive a closed form expression for the solution.

b) Calculate an expression for the gradient descent update using step size $\mu \geq 0$.

c) Calculate an expression for the update of gradient descent with line search.

d) Calculate an expression for the conjugate gradient update.

e) Calculate an expression for the coordinate descent update.

8. Consider the optimization problem

$$\hat{x} = \arg\min_{x \in \mathbb{R}^N} \left\{ \| y - Ax \|^2_\Lambda + x^t B x \right\}$$

where $A$ is a nonsingular $N \times N$ matrix, $B$ is a symmetric positive-definite $N \times N$ matrix, and $\Lambda$ is a diagonal positive-definite matrix. Furthermore, let $x^{(k)}$ denote the approximate solution of the optimization after $k$ iterations of an optimization technique, and let $\epsilon^{(k)} = x^{(k)} - \hat{x}$ be the convergence error after $k$ iterations.

a) Calculate the update recursion for $\epsilon^{(k)}$ with gradient descent optimization.

b) Under what conditions does gradient descent have stable convergence?

c) Calculate the update recursion for $\epsilon^{(k)}$ with ICD optimization. (Assume a single iteration consists of a single update of each pixel in raster order)
d) Under what conditions does coordinate descent have stable convergence?
Chapter 6

Non-Gaussian MRF Models

In Chapters 4 and 5, we introduced the GMRF and showed how it can be used as an image model in applications such as image restoration or reconstruction. However, one major limitation of the GMRF is that it does not accurately model the edges and other discontinuities that often occur in real images. In order to overcome this limitation, we must extend our approach to non-Gaussian models.

The purpose of this chapter is to generalize the GMRF of the previous chapters to non-Gaussian random fields. To do this, we will introduce the concept of a pair-wise Gibbs distribution, and show that, while it includes the GMRF, it can be used to naturally generalize the GMRF to non-Gaussian random fields. We will also present a set of commonly used non-Gaussian MRF models, and develop some tools for selecting their parameters.

6.1 Continuous MRFs Based on Pair-Wise Cliques

In this section, we develop image models which explicitly represent the probability distribution for an image in terms of the differences between neighboring pixels. In order to do this, we must first reformulate the GMRF model so that it is expressed in terms of pixel differences. Once this is done, we can then generalize the GMRF model to non-Gaussian distributions.

First, let us review the concept of a neighborhood system, $\partial s$, and its associated pair-wise cliques, $\mathcal{P}$, from Section 4.6. Recall that $\partial s \subset S$ is the set of neighboring pixels to $s$ and that $r \in \partial s$ if and only if $s \in \partial r$. So if $s$ is a neighbor of $r$, then $r$ must be a neighbor of $s$. Then for a given neighborhood
system, we defined the set of pair-wise cliques as $\mathcal{P} = \{ \{ s, r \} : s \in \partial r \}$. So $\mathcal{P}$ is then the set of all unordered neighboring pixel pairs $\{ s, r \}$ such that $r \in \partial s$.

Using these conventions, the distribution of a zero-mean GMRF can be written as

$$
p(x) = \frac{1}{z} \exp \left\{ -\frac{1}{2} x^t B x \right\},
$$

(6.1)

where $B_{r,s} = 0$ when $r \not\in \partial s \cup \{s\}$, and $z$ is the normalizing constant for the distribution known as the partition function. We also note that $X$ is then a Gaussian random vector with density $N(0, B^{-1})$.

We would like to rewrite this expression in a way that makes the dependence on differences between neighboring pixels more explicit. To do this, we introduce a very useful vector-matrix relationship as the following property.

**Property 6.1. Pairwise quadratic form identity** - Let $B \in \mathbb{R}^{N \times N}$ be any symmetric matrix and let $x \in \mathbb{R}^N$ be any vector. Then the following identity holds

$$
x^t B x = \sum_{s \in S} a_s x_s^2 + \frac{1}{2} \sum_{s \in S} \sum_{r \in S} b_{s,r} |x_s - x_r|^2,
$$

(6.2)

where $a_s = \sum_{r \in S} B_{s,r}$ and $b_{s,r} = -B_{s,r}$.

Moreover, for the particular case of a GMRF with neighborhood system $\partial s$, we know that the terms $B_{s,r} = 0$ for $r \not\in \partial s \cup \{s\}$, so we have the following general relationship.

$$
x^t B x = \sum_{s \in S} a_s x_s^2 + \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} |x_s - x_r|^2
$$

(6.3)

Notice that in the case of (6.3) the factor of $1/2$ is removed because the sum is over all unique unordered pairs of $\{ s, r \}$.

When used to model images, the coefficients $a_s$ are most often chosen to be zero. By choosing $a_s = 0$, we ensure that the prior probability of an image $x$ is invariant to additive constant shifts in the pixel values. To see this, consider the image $x$ and a shifted image, $\tilde{x}_s = x_s + 1$ for all $s \in S$. If $a_s = 0$, then only terms of the form $x_s - x_r$ remain in the expression of (6.2). Since any additive constant cancels in these terms, we know that

$$
\tilde{x}^t B \tilde{x} = x^t B x,
$$
so the two images are equally probable.\footnote{However, it should be noted that in this case the determinant of the inverse covariance matrix $B$ becomes zero, which is technically not allowed. However, this technical problem can always be resolved by computing the MAP estimate with $a_s > 0$ and then allowing $a_s$ to go to zero.}

By dropping these terms, and using the relationship of (6.3), we get the following **pair-wise GMRF distribution**:\footnote{Again, we note that the value of the partition function, $z$, in this expression becomes infinite because of the singular eigenvalue in $B$. However, this technical problem can be resolved by taking the limit as $a_s \to 0$.}

$$p(x) = \frac{1}{z} \exp \left\{ - \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} \frac{|x_s - x_r|^2}{2} \right\}$$ \hspace{1cm} (6.4)

Notice that the distribution of (6.4) is explicitly written in terms of the differences between neighboring pixels. So if $b_{s,r} > 0$, then as the squared pixel difference $|x_s - x_r|^2$ increases, the probability decreases. This is reasonable since we would expect nearby pixels to have similar values.

However, a limitation of the GMRF model is that it can excessively penalize the differences between neighboring pixels. This is because, in practice, the value of $|x_s - x_r|^2$ can become very large when $x_s$ and $x_r$ fall across a discontinuous boundary in an image. A simple solution to this problem is to replace the function $\frac{|x_s - x_r|^2}{2}$ with a new function $\rho(x_s - x_r)$, which grows less rapidly. With this strategy in mind, we define the **pair-wise Gibbs distribution** as any distribution with the form

$$p(x) = \frac{1}{z} \exp \left\{ - \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} \rho(x_s - x_r) \right\}$$ \hspace{1cm} (6.5)

Later in Chapter 14, we will give a more general definition for the Gibbs distributions, but it is interesting to note that the concept is borrowed from statistical thermodynamics (See Appendix B). In thermodynamics, the Gibbs distribution is a distribution of the form $p(x) = \frac{1}{Z} \exp \{-u(x)\}$ where $u(x)$ is the energy of the system.\footnote{In thermodynamics, the more general expression for the Gibbs distribution (also known as the Boltzmann distribution) is $p(x) = \frac{1}{Z} \exp \left\{ -\frac{1}{k_bT} u(x) \right\}$ where $k_b$ is Boltzmann’s constant, $T$ is the absolute temperature, and $u(x)$ is the energy of the system in state $x$. So we will assume that $k_bT = 1$.} So for our problem, the **energy function** of the pair-wise Gibbs distribution is given by

$$u(x) = \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} \rho(x_s - x_r)$$ \hspace{1cm} (6.6)
where the function $\rho(\Delta)$ for $\Delta = x_s - x_r$ is known as the potential function, and its derivative, $\rho'(\Delta) = \frac{d\rho(\Delta)}{d\Delta}$, is known as the associated influence function.

The choice of the potential function, $\rho(\Delta)$, is very important since it determines the accuracy of the MRF model, and ultimately in applications, it can determine the accuracy of the MAP estimate. While the selection of the potential function will be a major theme of this section, we can immediately introduce three properties that we will expect of $\rho(\Delta)$.

Zero at origin: $\rho(0) = 0$

Symmetric: $\rho(-\Delta) = \rho(\Delta)$

Monotone increasing: $\rho(|\Delta| + \epsilon) \geq \rho(|\Delta|)$ for all $\epsilon > 0$

Importantly, the pair-wise Gibbs distribution of equation (6.5) is still the distribution of an MRF. That is, each pixel, $x_s$, is conditionally independent of the remaining pixels given its neighbors. To see this, we can factor the pair-wise Gibbs distribution into two portions: the portion that has a dependency on $x_s$, and the portion that does not. This results in the relation,

$$p(x) = \frac{1}{z} \exp \left\{ - \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} \rho(x_s - x_r) \right\}$$

$$= \frac{1}{z} \exp \left\{ - \sum_{r \in \partial s} b_{s,r} \rho(x_s - x_r) \right\} f(x_{r\neq s}),$$

where $f(x_{r\neq s})$ is a function of the pixels $x_r$ for $r \neq s$. Using the factored form, we may calculate the conditional distribution of $X_s$ given the remaining pixels $X_r$ for $r \neq s$ as

$$p_{x_s|x_{r\neq s}}(x_s|x_{r\neq s}) = \frac{p(x)}{\int \mathbb{R} p(x_s, x_{r\neq s})dx_s} = \frac{p(x_s, x_{r\neq s})}{\int \mathbb{R} p(x_s, x_{r\neq s})dx_s}$$

$$= \frac{\frac{1}{z} \exp \left\{ - \sum_{r \in \partial s} b_{s,r} \rho(x_s - x_r) \right\} f(x_{r\neq s})}{\int \mathbb{R} \frac{1}{z} \exp \left\{ - \sum_{r \in \partial s} b_{s,r} \rho(x_s - x_r) \right\} f(x_{r\neq s})dx_s}$$

$$= \frac{\exp \left\{ - \sum_{r \in \partial s} b_{s,r} \rho(x_s - x_r) \right\}}{\int \mathbb{R} \exp \left\{ - \sum_{r \in \partial s} b_{s,r} \rho(x_s - x_r) \right\} dx_s}.$$
However, since this result is only a function of \( x_s \) and its neighbors, we have that 

\[
p_{x_s|x_{r \neq s}}(x_s|x_{r \neq s}) = p_{x_s|x_\partial s}(x_s|x_\partial s).
\]

This type of local dependency in the pixels of \( X \) is very valuable. In fact, it is valuable enough that it is worth giving it a name.

**Definition 9. Markov Random Field (MRF)**

A random field, \( X_s \) for \( s \in S \) with neighborhood system \( \partial s \) is said to be a Markov random field (MRF) if for all \( s \in S \), the conditional distribution of \( X_s \) is only dependent on its neighbors, i.e.,

\[
p_{x_s|x_{r \neq s}}(x_s|x_{r \neq s}) = p_{x_s|x_\partial s}(x_s|x_\partial s),
\]

where \( x_\partial s \) denotes the set of \( x_r \) such that \( r \in \partial s \).

Using this definition, we may state the following very important property of pair-wise Gibbs distributions.

**Property 6.2. Pair-wise Gibbs distributions are the distributions of an MRF**

- Let \( X_s \) have a pair-wise Gibbs distribution, \( p(x) \), with pair-wise cliques \( \mathcal{P} \), then \( X_s \) is an MRF with neighborhood system \( \partial s = \{ r \in S : \{ r, s \} \in \mathcal{P} \} \), and the conditional probability of a pixel given its neighbors is given by

\[
p_{x_s|x_\partial s}(x_s|x_\partial s) = \frac{1}{z} \exp \left\{ - \sum_{r \in \partial s} b_{s,r} \rho(x_s - x_r) \right\}, \tag{6.7}
\]

where \( z = \int_\mathbb{R} \exp \left\{ - \sum_{r \in \partial s} b_{s,r} \rho(x_s - x_r) \right\} dx_s \).

It is important to note, that while pair-wise Gibbs distributions parameterize a family of MRFs, there exist MRFs that do not have pair-wise Gibbs distributions. So the family of all distributions for MRFs is much larger. In fact, in Section 14.1, we will discuss this issue in detail, and show that the family of all MRFs is parameterized by a specific family of Gibbs distributions constructed from more general groups of pixels known as cliques.
6.2 Selection of Potential and Influence Functions

The question remains of how to select a potential function that best models real images? One place to start is to look at the conditional distribution of a pixel, $x_s$, given its neighbors, $x_{\partial s}$. In particular, the most probable value of $x_s$ given its neighbors is the solution to the optimization problem,

$$\hat{x}_s = \arg \max_{x_s \in \mathbb{R}} \log p_{x_s|x_{\partial s}}(x_s|x_{\partial s})$$

$$= \arg \min_{x_s \in \mathbb{R}} \sum_{r \in \partial s} b_{s,r} \rho(x_s - x_r) , \quad (6.8)$$

which can be in turn computed as the solution to the equation

$$\sum_{r \in \partial s} b_{s,r} \rho'(\hat{x}_s - x_r) = 0 . \quad (6.9)$$

Equation (6.8) has an interpretation of energy minimization where the terms $\rho(x_s - x_r)$ represent the potential energy associated with the two pixels $x_s$ and $x_r$. Alternatively, equation (6.9) has the interpretation of a force balance equation where the terms $\rho'(\hat{x}_s - x_r)$ represent the force that the pixel $x_r$ exerts on the estimate $\hat{x}_s$.

This interpretation of $\rho(x_s - x_r)$ as energy and $\rho'(x_s - x_r)$ as force serves as a useful tool for the design and selection of these potential functions. The function $\rho'(\Delta)$ is usually referred to as the influence function because it determines how much influence a pixel $x_r$ has on the MAP estimate of a pixel $\hat{x}_s$.

Figure 6.1 shows the potential and influence functions for two possible choices of the potential function which we will refer to as the Gaussian potential and Weak-spring potential \cite{11, 10} functions.

Gaussian potential: $\rho(\Delta) = |\Delta|^2$

Weak-spring potential: $\rho(\Delta) = \min \{|\Delta|^2, T^2\}$

Notice that the weak-spring potential requires the choice of an additional parameter $T$ that we set to 1 for the plots. Along with each potential function, we also show the associated influence function since this gives an indication of the “force” resulting from neighboring pixels. Notice that with the Gaussian prior, the influence of a pixel is linearly proportional to the value of $\Delta = |\Delta|^2$. \[ \hat{x}_s = \arg \max_{x_s \in \mathbb{R}} \log p_{x_s|x_{\partial s}}(x_s|x_{\partial s}) \]

$$= \arg \min_{x_s \in \mathbb{R}} \sum_{r \in \partial s} b_{s,r} \rho(x_s - x_r) , \quad (6.8)$$

which can be in turn computed as the solution to the equation

$$\sum_{r \in \partial s} b_{s,r} \rho'(\hat{x}_s - x_r) = 0 . \quad (6.9)$$

Equation (6.8) has an interpretation of energy minimization where the terms $\rho(x_s - x_r)$ represent the potential energy associated with the two pixels $x_s$ and $x_r$. Alternatively, equation (6.9) has the interpretation of a force balance equation where the terms $\rho'(\hat{x}_s - x_r)$ represent the force that the pixel $x_r$ exerts on the estimate $\hat{x}_s$.

This interpretation of $\rho(x_s - x_r)$ as energy and $\rho'(x_s - x_r)$ as force serves as a useful tool for the design and selection of these potential functions. The function $\rho'(\Delta)$ is usually referred to as the influence function because it determines how much influence a pixel $x_r$ has on the MAP estimate of a pixel $\hat{x}_s$.

Figure 6.1 shows the potential and influence functions for two possible choices of the potential function which we will refer to as the Gaussian potential and Weak-spring potential \cite{11, 10} functions.

Gaussian potential: $\rho(\Delta) = |\Delta|^2$

Weak-spring potential: $\rho(\Delta) = \min \{|\Delta|^2, T^2\}$

Notice that the weak-spring potential requires the choice of an additional parameter $T$ that we set to 1 for the plots. Along with each potential function, we also show the associated influence function since this gives an indication of the “force” resulting from neighboring pixels. Notice that with the Gaussian prior, the influence of a pixel is linearly proportional to the value of $\Delta = |\Delta|^2$. \[ \hat{x}_s = \arg \max_{x_s \in \mathbb{R}} \log p_{x_s|x_{\partial s}}(x_s|x_{\partial s}) \]
Figure 6.1: List of the potential and influence functions for the Gaussian prior and weak-spring model. With the Gaussian prior, the influence of a pixel is unbounded, and it increases linearly with the value of $\Delta$. However, with the weak spring model, the influence drops to zero when the value of $|\Delta|$ exceeds a threshold $T$.

$x_s - x_r$. Therefore, the influence of $x_r$ on the estimate of $\hat{x}_s$ is unbounded, and a neighboring pixel on the other side of an edge can have an unbounded effect on the final estimate of $\hat{x}_s$.

Alternatively, with the weak spring potential, the influence of a neighboring pixel is bounded because the potential function is clipped to the value $T^2$. This bounded influence of neighboring pixels is clearly shown by the associated influence function, which goes to zero for $|\Delta| > T$. This means that pixel differences greater than $T$ have no influence on the final estimate of $\hat{x}_s$. If the pixels lie across a boundary or edge in an image, this lack of influence can be very desirable, since it minimizes any blurring and therefore preserves the edge detail.

Figure 6.2 shows some additional potential functions with a similar na-
ture to the weak-spring potential. Each of these potential functions has the property that $\rho(\Delta)$ is a non-convex function of $\Delta$. (Appendix A contains a review of convexity including precise definitions and important properties.) In fact, all non-convex potential functions have the property that their influence functions (i.e., derivatives) must sometimes decrease as $\Delta$ increases.

Non-convex potential functions do a good job of precisely defining edges. Intuitively, non-convex functions typically have a region $|\Delta| < T_c$ such that $\rho'(\Delta)$ is strictly increasing (i.e., $\rho(\Delta)$ is locally convex), and a region $|\Delta| > T_c$ such that $\rho'(\Delta)$ is decreasing (i.e., $\rho(\Delta)$ is locally concave). These two regions clearly delineate between the “edge” and “non-edge” cases. However, in the next section, we will also see that non-convex cost functions have limitations.

<table>
<thead>
<tr>
<th>$\rho(\Delta)$</th>
<th>Reference</th>
<th>Potential Function</th>
<th>Influence Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\min{</td>
<td>\Delta</td>
<td>^2, T}$</td>
<td>Blake &amp; Zisserman</td>
</tr>
<tr>
<td>$\frac{\Delta^2}{T^2+\Delta^2}$</td>
<td>Geman &amp; McClure</td>
<td>Geman and McClure Potential</td>
<td>Geman and McClure Influence</td>
</tr>
<tr>
<td>$\log(T^2+\Delta^2)$</td>
<td>Hebert &amp; Leahy</td>
<td>Hebert and Leahy Potential</td>
<td>Hebert and Leahy Influence</td>
</tr>
<tr>
<td>$\frac{</td>
<td>\Delta</td>
<td>}{T+</td>
<td>\Delta</td>
</tr>
</tbody>
</table>

Figure 6.2: List of the potential and influence functions for a variety of non-convex potential functions plotted for $T = 1$. 
6.3 Convex Potential Functions

While non-convex potential functions can preserve edges better than a GMRF model, they also tend to generate discontinuous estimates at image boundaries which are typically perceived as undesirable visual artifacts. For some applications, such as edge detection, this type of abrupt change at edges can be desirable. However, in applications where the output needs to be viewed as a physical image (i.e., photography, medical imaging, etc.), then this type of very abrupt change can be undesirable.

Figures 6.3 illustrates how this discontinuous behavior can effect the result of MAP estimation. Figure 6.3(a) shows two slightly different signals that are used as input to a MAP cost function which uses the non-convex weak-spring prior model. (See [13] for details of the experiment and model.) Notice that only a slight change in the input, \( y \), results in a discontinuous jump in the MAP estimate shown in Figure 6.3(b). While this type of discontinuous behavior can be desirable in applications such as edge detection, it is typically not good when the result, \( x \), is to be viewed as an image, or when it is important to preserve subtle details of continuous variation in the original data.

![Noisy Signals](image1.png)

![Unstable Reconstructions](image2.png)

Figure 6.3: Illustration of how the MAP estimate can actually be unstable in practice. Figure a) shows two slightly different signals that are used as input to a MAP cost function incorporating non-convex potential functions. Notice that only a slight change in the input, \( y \), results in a discontinuous jump in the MAP estimate shown in Figure b).

In Section 7.1, we will see that convex potential functions have two major advantages when the data term is also convex. First, they result in contin-
uous or stable MAP estimates, so the resulting images tend to have fewer undesirable artifacts. And second, the resulting optimization problem for MAP estimation is also convex, so it is typically possible to compute the global maximum of the posterior density in tractable time.

<table>
<thead>
<tr>
<th>$\rho(\Delta)$</th>
<th>Reference</th>
<th>Potential Function</th>
<th>Influence Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta^2/2$</td>
<td>Gaussian MRF</td>
<td>“Quadratic”</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>\Delta</td>
<td>$</td>
<td>Besag [9]</td>
</tr>
<tr>
<td>$\begin{cases} \Delta^2/2 &amp; \text{for }</td>
<td>\Delta</td>
<td>&lt; T \ T</td>
<td>\Delta</td>
</tr>
<tr>
<td>$T^2 \log \cosh \left{ \frac{</td>
<td>\Delta</td>
<td>}{T} \right}$</td>
<td>Green [32]</td>
</tr>
<tr>
<td>$\frac{</td>
<td>\Delta</td>
<td>^p}{p} \left( 1 + \frac{</td>
<td>\Delta/T</td>
</tr>
</tbody>
</table>

Figure 6.4: List of the potential and influence functions for a variety of convex potential functions for $T = 1$ and shape parameters $p = 1.2$ and $q = 2$.

Over the years, a wide variety of convex potential functions have been
proposed with a correspondingly wide range of practical and theoretical advantages and disadvantages to consider. Figure 6.4 shows a representative subset of these functions, although there are many more.

Of course, the simplest choice of potential function is the quadratic, \( \rho(\Delta) = \Delta^2 \), which results in the Gaussian MRF model that was the subject of Chapter 4. While this model has the advantage of being the most analytically tractable, its primary disadvantage is that it does not accurately characterize the sharp discontinuities that typically occur in images and other real signals.

One of the earliest and most interesting alternative potential functions is the absolute value function initially proposed by Besag [9], which is sometimes referred to as a \( L_1 \) norm regularizer. In this case, the potential function is given by

\[
\rho(\Delta) = |\Delta| ,
\]

with associated influence function

\[
\rho'(\Delta) = \text{sign}(\Delta) .
\]

This potential function has the advantages that it is convex; it does not grow as quickly as a quadratic function, so it is edge preserving; and it does not require the choice of a specific \( T \) parameter that determines the value of \( \Delta \) that will likely occur across edges.

The absolute value potential function represents a special case because it is convex, but not strictly convex; and because it has a discontinuous second derivative at 0. This results in many very unique properties to the MAP estimate. For example, because the MAP cost function is not continuously differentiable, gradient based algorithms such as gradient descent, conjugate gradient and ICD are generally not convergent when used with the absolute value potential function, and more complex optimization algorithms are required that often are not guaranteed to achieve the global minimum [58].

We should note that the absolute value potential function is very closely related to the total variation regularization of [57] in which image estimates are regularized using the integral of the gradient magnitude. The total variation norm is defined for continuous functions on \( \mathbb{R}^2 \) by

\[
TV(x) = \int_{\mathbb{R}^2} |\nabla x(r)|dr .
\]
But of course in applications, it is necessary to discretely sample the function \( x(r) \). So the TV norm can be discretely approximated, using a rectangular sampling lattice, by either the isotropic total variation given by

\[
TV(x) = \sum_{i,j} \sqrt{(x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2}, \tag{6.12}
\]

or the anisotropic total variation given by

\[
TV(x) = \sum_{i,j} |x_{i+1,j} - x_{i,j}| + |x_{i,j+1} - x_{i,j}|. \tag{6.13}
\]

Notice that the anisotropic total variation of (6.13) is a special case of the pair-wise Gibbs distribution with the absolute value potential function. However, as the name implies, the anisotropic total variation of (6.13) does not accurately approximate the total variation for all orientations of the gradient. Nonetheless, its simplified form sometimes has computational advantages.

Figure 6.5 illustrates what typically happens when absolute value potentials are used. Notice that the minimum of the cost function typically falls along these edges of discontinuity. Since the edges of discontinuity occur with \( |x_r - x_s| = 0 \) or equivalently, \( x_r = x_s \), the MAP estimate tends to contain groups of neighboring pixels that are of equal value. This means that the absolute value potential tends to produce large flat regions with sharp transitions. In some cases, this may be good, but in other cases, these flat regions can appear artificial and are often described as “waxy” by naive observers.

In some applications, this tendency of the absolute value potential function or \( L_1 \) norm log prior model to drive values to zero can be quite useful. For example, the least absolute shrinkage and selection operator (LASSO) uses this property of the \( L_1 \) norm to drive model parameter values to 0, thereby reducing model order [67].

However, in other applications, the \( L_1 \) norm’s tendency to zero out regularized quantities can result in image artifacts; so in these applications, a less severe regularizing potential function is needed. There are two basic approaches to addressing this limitation of absolute value prior models. One approach is to generalize the \( L_1 \) norm to an \( L_p \) norm, and a second approach is to “round out” the bottom of the \( L_1 \) norm by making it quadratic for \(|\Delta| << 1\).
Figure 6.5: Figure showing a typical MAP cost function resulting from the use of an absolute value potential function (i.e., total variation regularization). Notice that the cost function is convex, but that it is not continuously differentiable, so that the minimum of the cost function typically occurs at a point of discontinuity corresponding to $x_s = x_r$ where $r \in \partial s$.

An example of the first approach is the potential function used for the Generalized Gaussian MRF (GGMRF) [13]. In this case, the potential function is given by

$$\rho(\Delta) = \frac{|\Delta|^p}{p}, \tag{6.14}$$

with associated influence function

$$\rho'(\Delta) = |\Delta|^{p-1}\text{sign}(\Delta). \tag{6.15}$$

For $p > 1$ this potential function is strictly convex and continuously differentiable; so gradient based algorithms can achieve a global minimum of the cost function. In addition, these potential functions have the advantage that, as with the case of the absolute value potential, there is no longer an explicit value of $T$ to choose, but there is instead a general shape parameter $p$. In practice, $p$ is often chosen to be in the range of 1.1 to 1.2.

Another approach to generalizing the absolute value potential is to round out the point of the function by using a local quadratic near $\Delta = 0$. Perhaps the best example of this approach is the Huber function [65]. This potential function is given by

$$\rho(\Delta) = \begin{cases} \frac{\Delta^2}{2} & \text{for } |\Delta| < T \\ T|\Delta| - \frac{T^2}{2} & \text{for } |\Delta| \geq T \end{cases} \tag{6.16}$$
with associated influence function

$$\rho'(\Delta) = \text{clip}\{\Delta, [-T, T]\} , \quad (6.17)$$

where the function $\text{clip}\{x, [a, b]\}$ simply clips the value $x$ to the interval $[a, b]$. The Huber function has the advantage that near $\Delta = 0$ it maintains many of the properties of the GMRF. This can be very important because the images generated with Huber function potentials tend to better maintain low-contrast details, and they also result in MAP cost functions that are more amenable to optimization. However, the Huber function does have some disadvantages. First, it again requires the choice of a threshold $T$. Second, for values of $|\Delta| > T$, the function is not strictly convex, so the global minimum of the MAP cost function may not be unique; and finally, the function does not have a continuous second derivative at $|\Delta| = T$. This last disadvantage is easily addressed by a wide variety of alternative potential functions which are approximately quadratic at $\Delta = 0$, and approximately linear as $|\Delta| \to \infty$. One such example is the function

$$\rho(\Delta) = T^2 \log \cosh \left( \frac{|\Delta|}{T} \right) , \quad (6.18)$$

with associated influence function

$$\rho'(\Delta) = T \tanh \left( \frac{|\Delta|}{T} \right) , \quad (6.19)$$

where again the constant $T$ is introduced to control the placement of the transition from quadratic to linear behavior [32].

Finally, the Q-Generalized GMRF (QGGMRF) is a generalization of the GGMRF potential function that allows for parameters to control both the low-contrast character of the function when $\Delta \approx 0$, and the high-contrast character as $|\Delta| \to \infty$ [66]. The QGGMRF potential function is given by

$$\rho(\Delta) = \frac{|\Delta|^p}{p} \left( \frac{\frac{\Delta}{T}|^{q-p}}{1 + |\Delta|^q} \right) \quad (6.20)$$

and its associated influence function is given by

$$\rho'(\Delta) = |\Delta|^{p-1} \frac{|\Delta|^{q-p} \left( \frac{q}{p} + |\Delta|^q \right)}{\left( 1 + |\Delta|^q \right)^2} \text{sign}(\Delta) \quad (6.21)$$
where $p$ and $q$ are chosen so that $1 \leq p < q$ in order to ensure that the function is convex. Along high contrast edges, when $|\Delta| > T$, the shape of the QGGMRF is approximately the same as the GGMRF with $\rho(\Delta) \approx \frac{|\Delta|^p}{p}$. However, in low-contrast regions when $|\Delta| < T$, then the potential function is approximately given by $\rho(\Delta) \approx \frac{|\Delta|^q}{pT^q}$. In a typical application, one might choose $q = 2$ and $1 \leq p < q$. In this case, the potential function is quadratic near $\Delta = 0$ and is ensured to have a continuous first and second derivative. Notice that this is not the case for the function $|\Delta|^p$, which has an unbounded second derivative near $\Delta = 0$ when $1 \leq p < 2$. Later we will see that a continuous second derivative is a valuable property in optimization, particularly when using the majorization techniques of Chapter 8.

6.4 Scaling Parameters for Non-Gaussian MRFs

In practical problems, it is almost always necessary to scale the range of values for a MRF model. So for example, if an MRF, $X$, represent a conventional raster image, then the values, $X_s$, should fall in a range from 0 to 255. However, if $X$ is being used to represent ocean depth in meters, then a range of 0 to 11,000 might be more appropriate. So for each physical phenomena, we must be able to scale the typical range of values in a MRF model, without necessarily changing the basic characteristics of the model.

To do this, we can introduce a scaling parameter, $\sigma_x$, to the pair-wise Gibbs distribution of equation (6.5). We do this by directly scaling the values inside the potential functions of the Gibbs distribution.

$$p(x|\sigma_x) = \frac{1}{z} \exp \left\{ - \sum_{(s,r) \in P} b_{s,r} \rho \left( \frac{x_s - x_r}{\sigma_x} \right) \right\}$$

Using this convention, notice that if $X$ has distribution $p(x|\sigma_x)$, then $\tilde{X} = X/\sigma_x$ must have a distribution given by $p(x|\sigma_x = 1)$.

Even though parameter estimation for MRFs is known to be a classically difficult problem, estimation of the scale parameter, $\sigma_x$, is more tractable than one might imagine. This is because the partition function, $z$, turns

---

4 Our definition of the QGGMRF swaps the roles for the variables $q$ and $p$ relative to the form used in [66].
out to be a simple function of $\sigma_x$. In fact, by applying standard results from multidimensional integration, it can be shown (See problem 5) that the partition function must have the form
\[
 z(\sigma) = z_0 \sigma^N,
\]
where $z_0$ is a constant. Using this result, we can express the ML estimate of $\sigma_x$ as
\[
 \hat{\sigma}_x = \arg \min_{\sigma_x > 0} \left\{ u(x/\sigma_x) + N \log(\sigma_x) + \log(z_0) \right\},
\]
where $u(x)$ is the Gibbs distribution’s energy function given by
\[
 u(x) = \sum_{\{s,r\} \in P} b_{s,r} \rho(x_s - x_r).
\]
By setting the derivative to zero, we can express the ML estimate as the solution to the following equation.
\[
 \frac{\sigma_x}{N} \frac{d}{d\sigma_x} u(x/\sigma_x) \bigg|_{\sigma_x = \hat{\sigma}_x} = -1 \quad (6.22)
\]
In fact, this equation is only a function of a single variable, $\sigma_x$; so it may be solved numerically for a particular image $x$ by using any rooting algorithm.

However, in some cases, it is even possible to find a closed form solution to the ML estimate. Consider the case when $u(\alpha x) = \alpha^p u(x)$, then it can be shown that for this case (See problem 6) the ML estimate has the closed form given by
\[
 \hat{\sigma}_x = \left( \frac{p}{N} u(x) \right)^{(1/p)}.
\]
For the specific case of the GGMRF of equation (6.14), the distribution is then given by
\[
 p(x|\sigma_x) = \frac{1}{z} \exp \left\{ -\frac{1}{p\sigma_x^p} \sum_{\{s,r\} \in P} b_{s,r} |x_s - x_r|^p \right\};
\]
so the ML estimate of $\sigma_x$ has the closed form given by
\[
 \hat{\sigma}_x^p = \frac{1}{N} \sum_{\{s,r\} \in P} b_{s,r} |x_s - x_r|^p. \quad (6.23)
\]
### 6.4 Scaling Parameters for Non-Gaussian MRFs

<table>
<thead>
<tr>
<th>Name</th>
<th>Potential Function $\rho(\Delta)$</th>
<th>Influence Function $\rho'(\Delta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quadratic</td>
<td>$\frac{\Delta^2}{2\sigma_x^2}$</td>
<td>$\frac{\Delta}{\sigma_x^2}$</td>
</tr>
<tr>
<td>Total Variation</td>
<td>$\frac{</td>
<td>\Delta</td>
</tr>
<tr>
<td>Huber</td>
<td>$\begin{cases} \frac{\Delta^2}{2\sigma_x^2} &amp; \text{for }</td>
<td>\Delta</td>
</tr>
<tr>
<td>log cosh</td>
<td>$T^2 \log \cosh \left( \frac{\Delta}{T\sigma_x} \right)$</td>
<td>$\frac{T}{\sigma_x} \tanh \left( \frac{\Delta}{T\sigma_x} \right)$</td>
</tr>
<tr>
<td>GGMRF</td>
<td>$\frac{</td>
<td>\Delta</td>
</tr>
<tr>
<td>QGGMRF</td>
<td>$\frac{</td>
<td>\Delta</td>
</tr>
</tbody>
</table>

Table 6.1: Analytical expressions for a number of convex potential functions and their associated influence function. Each potential function is parameterized by a scale parameter, $\sigma_x$, and a threshold parameter, $T$. The scale parameter changes the range of values in $X$, and the threshold parameter determines the point of transition in the potential functions.

Notice, that the very simple form of equation (6.23) is reminiscent of the expression for the variance of a Gaussian random variable, except that the sum is over all cliques and the differences are raised to the power $p$ rather than simply squared.

The setting of the single parameter $\sigma_x$ can be a major practical concern when using $p(x)$ as a prior model in, say, MAP estimation. If $\sigma_x$ is too large, then the image will be “under regularized” or in other words, it will over-fit the measured data and have too much noisy variation. Alternatively, if $\sigma_x$ is too small, the image will be “over regularized” and important detail may be removed due to over smoothing.

For notational simplicity, we can just include this parameter, $\sigma_x$, in the definition of the potential function. So for example, in the quadratic case the potential function becomes

$$
\rho(\Delta) = \frac{1}{2} \left| \frac{\Delta}{\sigma_x} \right|^2 = \frac{\Delta^2}{2\sigma_x^2}
$$
and its associated influence function becomes

\[ \rho'(\Delta) = \frac{\Delta}{\sigma_x^2}. \]

Using this approach, Table 6.1 lists out the potential and influence functions for a number of the convex potential functions discussed in the previous section. Notice, that the transition point between the low and high contrast behaviors now occurs at the value \( \sigma_x T \), rather than just \( T \). This is useful since it means that the units of \( T \) are automatically scaled to the physical range of the data based on the choice of \( \sigma_x \).
6.5 Chapter Problems

1. Prove Property [6.1] by showing that equation (6.2) holds for any symmetric matrix $B \in \mathbb{R}^{N \times N}$ and any vector $x \in \mathbb{R}^N$. (Hint: It is enough to show that the equality holds for $x = 0$ and that the gradients of the two expressions are equal for all $x$.)

2. Let $B$ be the precision matrix (i.e., the inverse covariance) of a zero-mean GMRF with pair-wise cliques $\mathcal{P}$. Then prove that equation (6.3) holds for any vector $x \in \mathbb{R}^N$. (Hint: Prove that equations (6.2) and (6.3) are equal by showing the following three facts:

   For all $\{s, r\} \notin \mathcal{P}$, $B_{s,r} = 0$.
   For $s = r$, the term $b_{s,r}|x_s - x_r|^2 = 0$.
   For all $s \neq r$, the term $b_{s,r}|x_s - x_r|^2$ appears twice in the sum of (6.2) and only once in the sum of (6.3).

3. Show that if $X$ is a zero-mean Gaussian random vector with inverse covariance, $B$, then $B_{s,r} = 0$ if and only if $X_s$ is conditionally independent of $X_r$ given $\{X_i: \text{for } i \neq s, r\}$.

4. Show that if $X$ is distributed as equation (6.5), then the conditional distribution of $X_s$ given $X_r$ for $r \neq s$ is given by equation (6.7).

5. Let $X \in \mathbb{R}^N$ have a Gibbs distribution with the form

$$p(x|\sigma) = \frac{1}{z} \exp \left\{- \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} \rho \left( \frac{x_s - x_r}{\sigma} \right) \right\}.$$

Then prove that the partition function is given by

$$z(\sigma) = \int_{\mathbb{R}^N} \exp \left\{- \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} \rho \left( \frac{x_s - x_r}{\sigma} \right) \right\} \, dx = z_0 \sigma^N,$$

where $z_0$ is a constant.
6. Prove that for a Gibbs distribution with the form

\[ p(x|\sigma) = \frac{1}{z} \exp \{ -u(x/\sigma) \} \]

with an energy function \( u(x) \) of the form

\[ u(x/\sigma) = \frac{1}{p\sigma^p} \sum_{\{s,r\} \in P} b_{s,r} |x_s - x_r|^p \]

then the ML estimate of the parameter \( \sigma \) is given by

\[ \hat{\sigma}^p = \frac{1}{N} \sum_{\{s,r\} \in P} b_{s,r} |x_s - x_r|^p \].
Chapter 7

MAP Estimation with Non-Gaussian Priors

The non-Gaussian prior models introduced in the previous chapter have the potential to improve image quality in applications such as image restoration and reconstruction. However, these non-Gaussian priors also require new methods for non-quadratic optimization so that the MAP estimate can be computed.

In this chapter, we introduce some of the basic tools that are commonly used for computing the solution to the non-quadratic optimization problems resulting from MAP estimation with non-Gaussian models. First, we will see that convexity is a powerful tool for ensuring that the MAP estimation problem has a unique and computable solution, and we will begin to understand the importance of efficient line-search algorithms when solving the large convex optimization problems that often arise in MAP optimization. However, even with these tools, we will see that computing updates for algorithms such as gradient descent, ICD, and conjugate gradient can be numerically demanding, which will lead us to the majorization minimization algorithms presented in Chapter 8.

7.1 The MAP Cost Function and Its Properties

In a typical inverse application, our objective is to estimate an image, $X \in \mathbb{R}^N$, with prior distribution $p(x)$ from a vector of measurements, $Y \in \mathbb{R}^M$, with conditional density $p_{y|x}(y|x)$. It is also very common to have physical
properties that restrict the image to a convex set $\Omega \subset \mathbb{R}^N$. For example, in many physical problems we know that the solution must be non-negative, so we would like to enforce a **positivity constraint** when solving the problem. To do this, we first define the set $\mathbb{R}^+ = \{x \in \mathbb{R} : x \geq 0\}$, and then the convex set is given by

$$\Omega = \mathbb{R}^{+N} = \{x \in \mathbb{R}^N : x_i \geq 0 \text{ for all } i\} \quad (7.1)$$

Using this framework, the MAP estimate of $X$ given $Y$ is given by

$$\hat{x} = \arg \min_{x \in \Omega} f(x; y) \quad (7.2)$$

where we will refer to the function $f(x; y)$ as the **MAP cost function** given by

$$f(x; y) = -\log p_{y|x}(y|x) - \log p(x) \quad (7.3)$$

So the three key steps required for MAP estimation are:

1. Select the forward model $p_{y|x}(y|x)$;
2. Select the prior model $p(x)$;
3. Minimize the cost function.

In principle, only steps 1 and 2 determine the quality of the MAP estimate, and the choice of optimization procedure only effects the computation required to compute the estimate. However, in practice if the optimization method does not achieve the exact global minimum of the MAP cost function (which is typically the case), then step 3 will also have some impact on the quality of the solution.

Convexity of the MAP cost function, $f(x; y)$, as a function of $x$ will be a very powerful concept in understanding the existence, uniqueness, and stability of the MAP estimate. Appendix A provides a review of convex functions and sets, and it also provides some basic results on how convexity can be used to ensure existence and uniqueness of minima in general, and the MAP estimate in particular. Loosely speaking, the key results are that, for a convex function, any local minimum must also be a global minimum; and for a strictly convex function, any global minimum must be unique.

---

1Existence, uniqueness, and stability are the three properties of a well-posed inverse as defined by Tikhonov [68].
7.1 The MAP Cost Function and Its Properties

Figure 7.1: Illustration of how a non-convex MAP cost function can lead to an unstable MAP estimate. Figures a) and b) show very slight perturbations of the MAP cost function corresponding to slightly different values of $y$. While the functions only vary slightly, the MAP estimate varies discontinuously jumping from one side to the other.

This is very important because most practical optimization techniques, such as the gradient-based methods of Chapter 5, converge at best to a local minimum of a cost function. So convexity of the MAP cost function ensures that practical optimization methods can result in a true (but perhaps non-unique) MAP estimate. Strict convexity further ensures that the resulting MAP estimate is unique.

While the MAP estimate may exist, it may not always have desirable properties. For example, Figure 6.3 of Section 6.3 provides an example where the MAP estimate behaves discontinuously. In some applications, this discontinuous behavior may be desirable, but in many applications, where $X$ is an image that must be restored, discontinuous behavior of the MAP estimate can introduce artifacts into the image.

Figure 7.1 graphically illustrates the problem that can occur with non-convex potential functions. Figures 7.1 a) and b) illustrate the two MAP cost functions corresponding to two values of $y$ that are only slightly different. While the MAP cost functions shift only slightly, the locations of the global minimum for each function can change discontinuously. This causes the MAP estimate to discontinuously jump from one side to the other. Intuitively, this is the reason for the discontinuous change in the MAP estimate as shown in Figure 6.3.

A strictly convex MAP cost function eliminates this potentially undesirable behavior by insuring that the MAP estimate is a continuous function of
the data \( y \). The following theorem makes this statement more precise [13].

**Theorem 7.1.1. Stability/Continuity of MAP Estimates**

Let the MAP cost function, \( f(x; y) \), be a continuous function of \((x, y) \in \Omega \times \mathbb{R}^M; \) and for all \( y \in \mathbb{R}^M \), let \( f(\cdot, y) \) be a strictly convex function of \( x \in \Omega \) with a local minimum. Then the MAP estimate is a unique and continuous (i.e., stable) function of the data \( y \).

This raises the question of what conditions will ensure convexity of the MAP cost function? In order to provide a practical answer to this question, we will consider a slightly more general example than the one considered in Chapter 5. Let us assume that \( Y \in \mathbb{R}^M \) has the form

\[
Y = AX + W ,
\]  

(7.4)

where \( W \in \mathbb{R}^M \) is assumed to be independent zero-mean additive Gaussian noise with distribution \( N(0, \Lambda^{-1}) \), \( X \in \mathbb{R}^N \), and \( A \) is an \( M \times N \) matrix. Typically, the inverse covariance \( \Lambda \) is diagonal because the noise for each scalar observation is independent; however, this is not absolutely necessary for the formulation of our problem. In addition, it may happen that \( M > N \), \( M < N \), or \( M = N \) depending on the specific experimental scenario.

Given this framework, the likelihood of the observations, \( Y \), given the unknown, \( X \), has a form similar to equation (5.3).

\[
p_{y|x}(y|x) = \frac{1}{(2\pi)^{M/2} |\Lambda|^{1/2}} \exp \left\{- \frac{1}{2} ||y - Ax||^2_{\Lambda} \right\}
\]  

(7.5)

In addition, we assume that \( X \) is an MRF, so that the prior is given by a pair-wise Gibbs distribution similar to that of equation (6.5),

\[
p(x) = \frac{1}{z} \exp \left\{- \sum_{\{s,r\} \in P} b_{s,r} \rho(x_s - x_r) \right\} ,
\]  

(7.6)

where in this case the potential function \( \rho(\Delta) \) includes the scaling constant \( \sigma_x \) as shown in Table 6.1 of Section 6.4. Putting the forward and prior models into equation (7.3), we get the MAP cost function of

\[
f(x; y) = \frac{1}{2} ||y - Ax||^2_{\Lambda} + \sum_{\{s,r\} \in P} b_{s,r} \rho(x_s - x_r) .
\]  

(7.7)
7.2 General Approaches to Optimization

From the properties of convex functions given in Appendix A, it is clear that if $\rho(\Delta)$ is a convex function, then $f(x; y)$ is a convex function of $x \in \Omega$. Moreover, if the rank of $A$ is at least $N$, then the function being minimized is strictly convex. However, even when $A$ is not rank $N$, the function will typically be strictly convex as long as $A1 \neq 0$ where $1$ denotes a column vector of 1’s. The following property makes this statement more precise.

Property 7.1. Conditions for strict convexity of the MAP cost function - Let $f(x; y)$ be defined as in (7.21) where $x \in \Omega \subset \mathbb{R}^N$, $y \in \mathbb{R}^M$, $\Lambda$ is a positive-definite matrix, $b_{s,r} \geq 0$ for all $s, r \in S$, and $\Omega$ is a convex set. Then $f(x; y)$ is a strictly convex function of $x \in \Omega$ and a continuous function of $(x, y) \in \Omega \times \mathbb{R}^M$ when either one of the following two conditions hold.

Condition 1: $A$ has rank $N$; and $\rho(\Delta)$ is convex on $\mathbb{R}$

Condition 2: $A1 \neq 0$; $\forall s \ \exists r$ s.t. $b_{s,r} > 0$; and $\rho(\Delta)$ is strictly convex on $\mathbb{R}$.

In practice, the second condition of Property 7.1 typically holds in applications as long as the potential functions are strictly convex.

7.2 General Approaches to Optimization

Once we have selected appropriate forward and prior models, the challenge becomes to accurately compute the MAP estimate through computationally efficiently minimization of the MAP cost function.

$$\hat{x} = \arg \min_{x \in \Omega} f(x)$$ (7.8)

In this chapter, we will be primarily concerned with the problem of convex optimization in which $\Omega$ is a closed convex set and $f(x)$ is a convex function. Then from Theorem A.0.2 of Appendix A we know that any local minimum will be a global minimum. Moreover, if the function $f(x)$ is strictly convex, than any such global minimum will be unique.

Now it may happen that no local or global minimum exists for the problem of (7.8). For example, Figure A.1 shows a simple function $f(x) = e^{-x}$ that has no local minimum on $\mathbb{R}$. We can ensure the existence of a local minimum by adding the assumption that $\Omega$ is a compact set, i.e., that it is closed and
bounded. However, in many cases the MAP cost function will take values on \(\mathbb{R}^N\) or \(\mathbb{R}^+\), which are not bounded and therefore not compact. In order to handle these cases, we will typically assume that the function has a **compact sublevel set** as defined in Appendix A. In this case, there is an \(\alpha \in \mathbb{R}\) such that the sublevel set given by

\[ \mathcal{A}_\alpha = \{ x \in \mathbb{R}^N : f(x) \leq \alpha \} \]

is compact and non-empty. Using this definition, it is easily shown that any strictly convex function with a compact sublevel set has a unique global minimum. (See Problem 2)

In some cases, the set \(\Omega\) will enforce constraints that can make optimization more difficult. If \(\hat{x}\) lies in the interior of \(\Omega\) and \(f(x)\) is continuously differentiable, then the familiar condition for a minimum is that

\[ \nabla f(x)|_{x=\hat{x}} = 0 \, . \]

However, things get more complicated when \(\hat{x}\) lies on the boundary of \(\Omega\). In this case, the **Karush-Kuhn-Tucker (KKT) conditions** provide a generalization of this familiar condition. Moreover, in Chapter 9 we will introduce the theory of Lagrange multipliers as a general approach to solving constrained optimization problems, and we will see that these techniques can also be extremely useful in solving MAP optimization problems when \(f(x)\) is not differentiable everywhere.

The constraint that typically arises in real imaging problems is positivity. Physically, we often know that \(x_s\) must be non-negative because it represents a non-negative physical quantity such as light emission, photon absorption, or density. In practice, the positivity constraint can provide quite a bit of useful information by effectively reducing the number of unknown variables in a problem when they hit the constraint of zero. Fortunately, the set of positive images, \(x \in \mathbb{R}^+\), is a convex set; so the problem remains one of convex optimization.

With this background in mind, let us first consider the problem of unconstrained optimization when \(x \in \mathbb{R}^N\). Most iterative optimization algorithms start with an initial value \(x^{(k)}\) and then update the value in an attempt to minimize the function \(f(x)\). As we saw in Chapter 5, this update step typically has the form of a line search in which the function \(f(x)\) is searched.
General Optimization Approach:

Initialize $x^{(0)} \in \mathbb{R}^N$

For $k = 1$ to $K$

// Calculate direction $d \in \mathbb{R}^N$

$d \leftarrow \text{FavoriteOptimizationAlgorithm}(f(\cdot), x^{(k-1)})$

// Perform line search

$\alpha^\ast \leftarrow \arg\min_{\alpha \in \mathbb{R}} f(x^{(k-1)} + \alpha d)$

// Update solution

$x^{(k)} \leftarrow x^{(k-1)} + \alpha^\ast d$

Figure 7.2: This figure provides a pseudocode specification for a general class of non-quadratic optimization algorithms. The structure is used by many optimization algorithms including gradient descent with line search, ICD, and conjugate gradient.

along a line in the direction $d$ passing through the initial point $x^{(k)}$. The search line can be represented as

$$x = x^{(k-1)} + \alpha d$$

where $d \in \mathbb{R}^N$ is the chosen direction and $\alpha \in \mathbb{R}$ is a scalar that parameterizes the movement along the line. Algorithmically, the line search update is then given by

$$\alpha^\ast = \arg\min_{\alpha \in \mathbb{R}} f(x^{(k-1)} + \alpha d)$$

$$x^{(k)} = x^{(k-1)} + \alpha^\ast d$$

where the first step searches along a line, and the second step updates the value of $x$. We can also represent this line search operation as a function

$$x^{(k)} = \text{LineSearch}(x^{(k-1)}; d).$$

Our hope is that after repeated application of line search that we will converge to a local or global minimum of the function $f(x)$. We can make this thinking a bit more precise by defining the concept of a fixed point. We say that $x^\ast$ is a fixed point of the function $\text{LineSearch}(\cdot; d)$ if $x^\ast = \text{LineSearch}(x^\ast; d)$, i.e., the line-search operation returns its initial value. Then $x^\ast$ is a global minimum of $f(x)$ if and only if $x^\ast$ is a fixed point of the function $\text{LineSearch}(\cdot; d)$ for all directions $d \in \mathbb{R}^N$.

Figure 7.2 provides a pseudocode specification of this general optimization framework. First the direction $d$ is determined. Next, a line search is performed to find the value of $\alpha \in \mathbb{R}$ that minimizes the function $f(x^{(k)} + \alpha d)$. 
General ICD Optimization Approach:

For $K$ iterations {
    For each pixel $s \in S$ {
        $\alpha^* \leftarrow \arg \min_{\alpha \geq -x_s} f(x + \alpha \varepsilon_s)$
        $x_s \leftarrow x_s + \alpha^*$
    }
}

Figure 7.3: This figure provides a pseudocode specification for the general ICD or equivalently Gauss-Seidel update algorithm. One full iteration requires the sequential update of each pixel. ICD works well with positivity constraints, and when $f(x)$ is continuously differentiable and convex then ICD updates converge to a global minimum when it exists.

Once the optimum value $\alpha^*$ is determined, the new value of $x^{(k+1)}$ is computed, and the entire process is repeated until the desired level of convergence is achieved. In practice, line search is common to many optimization algorithms; and for non-quadratic optimization problems, it tends to represent a key computational burden in the overall optimization procedure. So efficient implementation of line search is very important.

In the case of the steepest descent algorithm of Section 5.3.3, the direction is chosen to be in the opposite direction of the gradient, $d = -\nabla f(x)$. In the conjugate gradient algorithm of Section 5.5, the negative gradient direction is adjusted to be conjugate to all previous update directions. The ICD optimization algorithm of Section 5.4 is an interesting special case in which each new update is taken to be in the direction of a single pixel. So in this case, $d = \varepsilon_s$, where $\varepsilon_s \in \mathbb{R}^N$ is a vector whose $s^{th}$ component is 1, and remaining components are 0.

The ICD algorithm, which is also known as Gauss-Seidel iterations, will be of particular value and interest to us. A single full ICD iteration consists of one update for each pixel in a fixed sequential order. So we may represent that in the following algorithmic form.

\[
\text{For } s \in S \quad \{ \\
    \alpha^* \leftarrow \arg \min_{\alpha \in \mathbb{R}} f(x + \alpha \varepsilon_s) \\
    x \leftarrow x + \alpha^* \varepsilon_s \\
\} \tag{7.10}
\]

\[
\alpha^* \leftarrow \arg \min_{\alpha \geq -x_s} f(x + \alpha \varepsilon_s) \\
\tag{7.11}
\]

\[
x \leftarrow x + \alpha^* \varepsilon_s \\
\]

\}
In fact, the ICD algorithm we have studied is guaranteed to converge to the global minimum under some fairly general conditions. The following theorem states this result more precisely.

**Theorem 7.2.1. ICD Convergence**

Let \( f : \Omega \to \mathbb{R} \) be a strictly convex and continuously differentiable function on the convex set \( \Omega \subset \mathbb{R}^N \). Then the ICD iterations of equation (7.10) converge to the global minimum of \( f \).

**Proof.** See for example Proposition 3.9 page 219 of \([5]\). \( \square \)

However, while this result appears to be strong, the limitation to continuously differentiable functions is a practical restriction. If the function is convex but not differentiable, then the ICD algorithm can routinely fail to converge. For example, when using the total variation regularization of Section 6.3, the MAP cost function is not continuously differentiable, so the ICD algorithm will typically stall before reaching the global minimum. The Problems 7 and 5 at the end of this chapter present two examples of the application of ICD to functions that are not continuously differentiable. In the first problem, ICD can be shown to converge to the global minimum, while in the second problem it stalls.

One important advantage of ICD is that positivity is easily enforced by simply restricting each update to take on a non-negative value. In this case, the ICD optimization procedure is given by

\[
x_s \leftarrow x_s + \arg \min_{\alpha \geq -x_s} f(x + \alpha \varepsilon_s)
\]

where \( \varepsilon_s \) is a vector which is 1 at its \( s^{th} \) coordinate and zero elsewhere. Figure 7.3 shows a pseudo-code specification of the general ICD optimization algorithm with positivity constraints.

If the function \( f(x) \) is continuously differentiable and convex, then the solution to the optimization problem with positivity constraints can be expressed using the KKT conditions. In this case, the KKT conditions reduce
to the simple form.

\[ \forall s \in S \text{ such that } x_s^* \neq 0 \quad \frac{\partial f(x^*)}{\partial x_s} = 0 \]

\[ \forall s \in S \text{ such that } x_s^* = 0 \quad \frac{\partial f(x^*)}{\partial x_s} \geq 0 . \]

So in this case, the function takes on its global minimum at \( x^* \) if and only if the derivative for each coordinate is either zero or positive, and it can only be positive for coordinates that fall on the boundary of the positivity constraint.

Using these simplified KKT constraints, it is easily shown that \( x^* \) achieves a global minimum of a convex function if and only if it is a fixed point of the ICD algorithm. So once again, the ICD algorithm is globally convergent for the minimization of continuously differentiable convex functions with positivity constraints. This is an important property because other methods, such as gradient decent have difficulty with positivity constraints.

In Chapter 9, we will treat the broader problem of constrained optimization in the framework of Lagrange multipliers, and we will introduce some general methods such as variable splitting and the ADMM algorithm that will provide convergent methods for solving a wide range of convex optimization problems.

### 7.3 Optimization of the MAP Cost Function

Now that we have presented some of the basic strategies behind optimization, consider how these ideas can be applied to the specific MAP cost function of equation (7.21). For this problem, the line search function has the form

\[
f(x + \alpha d) = \frac{1}{2} ||y - Ax - \alpha A d||_A^2 + \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} \rho(x_s - x_r + \alpha (d_s - d_r))
\]

\[
= \frac{1}{2} ||e - \alpha A d||_A^2 + \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} \rho(\Delta x_{s,r} + \alpha \Delta d_{s,r}) ,
\]  \hspace{1cm} (7.12)

where \( e \triangleq y - Ax \), \( \Delta x_{s,r} \triangleq x_s - x_r \), \( \Delta d_{s,r} \triangleq d_s - d_r \), and we suppress the dependence on the data, \( y \), for notational clarity. Now since, in this case,
the log likelihood term has a quadratic form, equation (7.12) can be more compactly expressed as a simple quadratic function of the scalar value $\alpha$. We can do this by simply computing the first two terms of the Taylor series expansion, which in this case, will be an exact representation of the true log likelihood. This results in the following simpler expression for the MAP cost function

$$f(x + \alpha d) = c + \theta_1 \alpha + \frac{1}{2} \theta_2 \alpha^2 + \sum_{\{s,r\} \in P} b_{s,r} \rho(\Delta x_{s,r} + \alpha \Delta d_{s,r})$$  \hspace{1cm} (7.13)$$

where

$$\theta_1 = -e^t \Lambda Ad$$  \hspace{1cm} (7.14)$$
$$\theta_2 = d^t A^t \Lambda Ad ,$$  \hspace{1cm} (7.15)$$

where $e = y - Ax$. So our objective is then to minimize this function over $\alpha$, and then to use this minimized value to update $x$. The optimal value of $\alpha$ is then given by

$$\alpha^* = \arg \min_{\alpha \in [a,b]} f(x + \alpha d)$$  \hspace{1cm} (7.16)$$

$$= \arg \min_{\alpha \in [a,b]} \left\{ \theta_1 \alpha + \frac{1}{2} \theta_2 \alpha^2 + \sum_{\{s,r\} \in P} b_{s,r} \rho(\Delta x_{s,r} + \alpha \Delta d_{s,r}) \right\}$$  \hspace{1cm} (7.17)$$

where $[a, b]$ is the interval over which $\alpha$ can vary. For example, in gradient descent $[a, b] = [0, \infty]$. However, in practice it is usually possible to find tighter limits which result in a bounded interval. Once the value of $\alpha^*$ is computed, then the value of $x$ is updated,

$$x \leftarrow x + \alpha^* d ,$$

and the entire optimization procedure is repeated until sufficient convergence is achieved.

The following example considers the special case of ICD optimization and derives an efficient algorithm for computing the required updates.

*Example 7.1.* In order to illustrate the concept of non-quadratic line search, consider the example of ICD updates with a non-quadratic log prior model.
Referring back to Section 5.4, we see that the update direction for ICD is given by $d = \varepsilon_s$, where $\varepsilon_s$ is a vector that is 1 for element $s$, and 0 otherwise. Using this fact, and the update equations of (7.14) and (7.15), we get the following values of $\theta_1$ and $\theta_2$,

$$
\theta_1 = -e^t \Lambda A \varepsilon_s = -e^t \Lambda A_{*,s}, \tag{7.18}
$$

$$
\theta_2 = \varepsilon_s^t A^t \Lambda A \varepsilon_s = A_{*,s}^t \Lambda A_{*,s}, \tag{7.19}
$$

where $A_{*,s}$ is the $s^{th}$ column of the matrix $A$.

Remember that for ICD, the value of $\alpha$ only changes the value of the one pixel $x_s$. So the only terms that need to be considered are terms that contain $x_s$ in them. Therefore, we have the following relationship.

$$
\sum_{\{s,r\} \in P} b_{s,r} \rho(\Delta x_{s,r} + \alpha \Delta d_{s,r}) = \sum_{r \in \partial s} b_{s,r} \rho(\Delta x_{s,r} + \alpha) + \text{constant}
$$

So using equation (7.17), we get the following ICD update equation for the $s^{th}$ pixel,

$$
\alpha^* = \arg \min_{\alpha \geq -x_s} \left\{ \theta_1 \alpha + \frac{1}{2} \theta_2 \alpha^2 + \sum_{r \in \partial s} b_{s,r} \rho(x_s - x_r + \alpha) \right\}, \tag{7.20}
$$

where the constraint that $\alpha \geq -x_s$ enforces that the updated pixel is positive. Of course, the optimization of equation (7.20) is not straightforward for most non-quadratic potential functions, $\rho(\cdot)$; but for now, let us assume that this 1-D optimization can be solved. The resulting value of $\alpha^*$ is then used to update both the pixel’s value and the error vector $e$ with the following update equations.

$$
x_s \leftarrow x_s + \alpha^*
$$

$$
e \leftarrow e - A \varepsilon_s \alpha^* = e - A_{*,s} \alpha^*
$$

Figure 7.4 provides a pseudocode summary of this procedure for ICD optimization with a non-quadratic prior model.

The question remains of how to efficiently compute the line search of equations (7.17) or (7.20). Notice that, in this form, the likelihood term is reduced to a very simple quadratic expression, so this is easily handled; but the prior
7.3 Optimization of the MAP Cost Function

ICD Algorithm for Non-Quadratic Prior:

Initialize $e \leftarrow y - Ax$

For $K$ iterations {

For each pixel $s \in S$ {

$\theta_1 \leftarrow -e^t \Lambda A_{*,s}$

$\theta_2 \leftarrow A^t_{*,s} \Lambda A_{*,s}$

$\alpha^* \leftarrow \arg \min_{\alpha \geq -x_s} \left\{ \theta_1 \alpha + \frac{1}{2} \theta_2 \alpha^2 + \sum_{r \in \partial s} b_{s,r} \rho'(\alpha + x_s - x_r) \right\}$

$x_s \leftarrow x_s + \alpha^*$

$e \leftarrow e - A_{*,s} \alpha^*$

}

}

Figure 7.4: Pseudocode for a fast implementation of the ICD algorithm with a non-quadratic prior term. The speedup depends on the use of a state variable to store the error, $e = y - Ax$. The resulting line search can be solved using a variety of rooting methods, such as half-interval search, or the surrogate function methods of Chapter 8.

term still depends on the non-quadratic potential functions. If the potential functions are continuously differentiable, then one simple way to find the minimum is to root the derivative of the function, in other words, to compute the value of $\alpha^*$ that makes the following equation true.

$$g(\alpha) \triangleq \frac{\partial}{\partial \alpha} f(x + \alpha d) = \theta_1 + \theta_2 \alpha + \sum_{\{s,r\} \in P} b_{s,r} \rho'(\Delta x_{s,r} + \alpha \Delta d_{s,r}) \Delta d_{s,r} = 0,$$

where $\rho'(\Delta)$ is the influence function formed by differentiating the potential function. If the potential functions are strictly convex, then $g(\alpha)$ will be a strictly increasing function of $\alpha$. So if we can find an interval so that $g(a) < 0$ and $g(b) > 0$, then the root must lie somewhere between these two limits.

There are many well known methods for efficiently rooting an equation of a scalar variable, but perhaps the most intuitive is the half interval search method. Figure 7.5 shows a pseudocode specification of half interval search. The idea is a simple one. With each iteration, $g(c)$ is evaluated where $c = (b + a)/2$ is the half way point between $a$ and $b$. Then if $g(c) \leq 0$, the point $c$ is taken as the new left hand end of the search interval; and if $g(c) > 0$, ...
Half Interval Line-Search Algorithm:

Initialize \( a \) and \( b \) so that \( g(a) < 0 \) and \( g(b) > 0 \).

For \( k = 1 \) to \( K \) {
  \( c \leftarrow (a + b)/2 \)
  If \( g(c) \leq 0 \), then \( a \leftarrow c \)
  If \( g(c) > 0 \), then \( b \leftarrow c \)
}\n
\[ \alpha^* = \frac{-g(a)}{g(b) - g(a)} (b - a) + a \]

Figure 7.5: This figure provides a pseudocode specification of half-interval line search. The algorithm requires an initial specification of the search interval, \([a, b]\), and then it iteratively halves the interval to reduce the uncertainty of the solutions.

then \( c \) is taken as the new right hand end of the interval. After \( K \) iterations, the algorithm is guaranteed to have achieved an accuracy of at least \( \frac{b-a}{2^K} \). Notice that the last step performs an interpolation of the function in order to estimate the location of the root. If the function is smooth, then this step can greatly improve the rooting accuracy.

However, half interval search has a number of disadvantages. First, it requires that a finite interval, \([a, b]\), for the solution be known in advance. There are a number of practical approaches to solving this problem. One approach is to first check the values of \( g(a) \) and \( g(b) \). If both \( g(a) < 0 \) and \( g(b) < 0 \), then \( a \leftarrow b \) can be taken as the new left-hand value, since it is the better of the two solutions, and the value of \( b \) can be increased. This process can then be repeated until \( g(b) > 0 \). A similar approach can be taken if \( g(a) > 0 \) and \( g(b) > 0 \). In this case, \( b \leftarrow a \) and \( a \) is decreased.

Perhaps a more serious disadvantage of half-interval search is that it requires repeated evaluation of the function \( g(\alpha) \). Typically, evaluation of \( g(\alpha) \) requires significant computation, so this may represent an unacceptable computational overhead. This overhead can be reduced by the judicious design of a line search algorithm which converges in a small number of iterations. Furthermore, it is important that with each iteration the value of the cost function, \( f(x + \alpha d) \), is monotonically decreased from its starting value, \( f(x) \). This ensures that, even if the iterations are terminated early, the cost function will still be reduced. The following chapter presents such an approach.
7.4 Chapter Problems

1. Consider the optimization problem

\[ \hat{x} = \arg \min_{x \in \mathbb{R}^N} \left\{ ||y - Ax||_\Lambda^2 + \lambda x^t Bx \right\} \]

where \( y \in \mathbb{R}^M, A \in \mathbb{R}^{M \times N}, \Lambda \) is a positive-definite \( M \times M \) matrix, and \( B \) is a positive-definite \( N \times N \) matrix, and \( \lambda > 0 \).

a) Show that the cost function is strictly convex.

b) Derive a closed form expression for the solution.

2. Theorem [A.0.5] of Appendix [A] requires that the function being minimized is strictly convex and that the domain of the function is compact. However, it is often the case that the domain of the function being optimized is not compact. The goal of this problem is to show that for some typical cases considered in this book, the global optimum for these problems still exists and is unique.

Consider the MAP cost function given by

\[ f(x) = \frac{1}{2} ||y - Ax||_\Lambda^2 + \sum_{\{s,r\} \in \mathcal{P}} b_{s,r}\rho(x_s - x_r) \]  \hspace{1cm} (7.21)

where \( y \in \mathbb{R}^N, x \in \mathbb{R}^N, A \in \mathbb{R}^{N \times N} \) has rank \( N, \Lambda \) is positive-definite, and \( \rho(\Delta) \) is a positive convex function of \( \Delta \). Also, define the sublevel set \( \mathcal{A}_\alpha \) to be

\[ \mathcal{A}_\alpha = \left\{ x \in \mathbb{R}^N : f(x) \leq \alpha \right\} , \]

and define the inverse image of a set \( S \subset \mathbb{R} \) to be

\[ f^{-1}(S) = \left\{ x \in \mathbb{R}^N : f(x) \in S \right\} . \]

For this problem you can use the following theorems:

T1: A set in \( \mathbb{R}^N \) is compact if and only if it is closed and bounded.

T2: If \( f \) is a continuous function and \( S \) is closed, then the inverse images \( f^{-1}(S) \) is closed.
a) Prove that for all $\alpha \in \mathbb{R}$ the sublevel set $\mathcal{A}_\alpha$ is closed.

b) Prove that there exists an $\alpha \in \mathbb{R}$ such that the sublevel set $\mathcal{A}_\alpha$ is non-empty and compact.

c) Prove there exists a MAP estimate, $\hat{x}$, so that $\forall x \in \mathbb{R}^N$, $f(\hat{x}) \leq f(x)$.

d) Prove that the MAP estimate is unique.

3. Show that if $f : \mathbb{R}^N \rightarrow \mathbb{R}$ is a strictly convex function, then $x^*$ achieves the global minimum of $f(x)$ if and only if $x^*$ is a fixed point of the function $\text{LineSearch}(\cdot; d)$ for all directions $d \in \mathbb{R}^N$.

4. Consider a cost function of the form

$$f(x) = \frac{1}{2}||y - Ax||_A^2 + \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} (x_s - x_r)^2.$$ 

Show that $f(x + \alpha d) = c + \theta_1 \alpha + \frac{1}{2} \theta_2 \alpha^2$ where

$$\theta_1 = -(y - Ax)^t \Lambda A d + \sum_{\{s,r\} \in \mathcal{P}} 2b_{s,r} (x_s - x_r) (d_s - d_r)$$ 

$$\theta_2 = d^t A^t \Lambda A d + \sum_{\{s,r\} \in \mathcal{P}} 2b_{s,r} (d_s - d_r)^2.$$ 

5. Define the function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ as

$$f(x) = |x_1 - x_0| + \frac{1}{8} |x_1 + x_0|^2$$

where $x = (x_0, x_1)$.

a) Prove that this function is convex.

b) Prove that the global minimum of the function is at $x^* = (0, 0)$.

c) Show that in general the ICD algorithm does not necessarily converge to the global minimum of $f(x)$.

6. Let the MAP cost function have the form of (7.21) where the rank of $A$ is equal to $N$, the potential functions are strictly convex, and the values $b_{s,r} \geq 0$. Show that the costs resulting from ICD updates form a monotone decreasing and convergent sequence.
7. Consider the MAP cost function given by

\[ f(x; y) = ||y - Ax||^2 + ||x||_1 \]

where \( x \) and \( y \) are vectors in \( \mathbb{R}^N \), \( ||x||_1 \) is the \( L_1 \) norm of \( x \), and \( A \) is a full rank matrix in \( \mathbb{R}^{N \times N} \).

a) Prove that \( f(x; y) \) is a strictly convex function of \( x \).
b) Calculate a closed form expression for the ICD update.
c) Prove that this function takes on a local minimum which is also its unique global minimum.
d) We say that, \( x^* \), is a fixed point of the ICD algorithm if a full ICD update using an initial value of \( x^* \) produces an updated value of \( x^* \). Using this definition, prove that \( x^* \) is a fixed point of the ICD algorithm if and only if it is a global minimum of \( f(x; y) \).
Chapter 8

Surrogate Functions and Majorization

As non-quadratic optimization has grown in importance, a very powerful set of methods have emerged known as majorization techniques, which allow non-quadratic optimization problems to be converted into iterative quadratic optimization. These methods go by a variety of names, but they all share a similar framework. The idea behind majorization techniques is to approximate the true function being minimized by a much simpler function known as a surrogate or substitute function.

In this chapter, we introduce the concepts of surrogate functions and majorization as powerful tools for efficiently computing the solution to the non-quadratic optimization problems resulting from MAP estimation with non-Gaussian models. We will see that surrogate functions can be very effective in a broad range of optimization problems because they allow for dramatic simplification of problems without sacrificing accuracy or convergence speed. Later in Chapter 12, we will also see that surrogate functions play a central role in understanding the EM algorithm.

\footnote{Over the years, roughly equivalent methods have been “rediscovered” for differing purposes, and many of these rediscoveries have resulted in new names for the method. One of the earliest references is the majorant technique of Kantorovich in [40] as a method for insuring the convergence of Newton iterations. Of course, the EM algorithm first introduced in its general form in [3, 2] implicitly uses this method for the iterative computation of the ML estimate. Ortega and Rheinboldt later discussed the method in the context of line search and referred to it as the “majorization principal” [50]. In [38], Huber used the concept of “global” curvature to minimize 1-D functions resulting from robust estimation procedures. This same concept was called “half-quadratic optimization” and applied to the problem of regularization or MAP estimation using non-quadratic priors by Geman in [27]. And, De Pierro introduced the concept to the image reconstruction community for the purposes of parallel optimization in tomographic reconstruction in [53]. The terms surrogate function [21] and substitute function [74] were later used to describe a technique for the parallelization of coordinate descent optimization in tomographic reconstruction.}
The central concept of majorization is to find a computationally efficient method for the minimization of a complicated scalar function $f(x)$, where $x \in \mathbb{R}^N$. Typically, the nature of the function $f$ makes it difficult to directly optimize, so our strategy will be to instead minimize a much simpler surrogate function denoted by $q$. Our surrogate function, $q$, will be designed to fit $f$ about a point of approximation denoted by $x'$ in much the same way that a Taylor series approximation is designed to fit a function near a point.

Figure 8.1 illustrates the situation. Both $f(x)$ and $q(x; x')$ are plotted as a function of $x$, and both functions are equal at the point of approximation, $x = x'$. Moreover, we will insist that $q(x; x')$ both equal $f(x)$ at $x = x'$ and that, in addition, $q(x; x')$ be an upper bound to $f(x)$ everywhere else. These two constraints can be expressed mathematically in the following two equations.

\[
\begin{align*}
  f(x') &= q(x'; x') \\
  f(x) &\leq q(x; x')
\end{align*}
\]  

(8.1)  

(8.2)

Importantly, we will see that these two conditions ensure that minimization of these surrogate function guarantees minimization of the true function. Again, Figure 8.1 illustrates the situation. Notice that in the figure $x^*$ is the global minimum of $f(x)$, while $\tilde{x}^*$ is the global minimum of the surrogate function $q(x; x')$. However, since the surrogate function must upper bound the true function, it must also be the case that $f(\tilde{x}^*) < f(x')$. So the result of
minimizing the surrogate function must also reduce the true function’s value.

Of course, we have not yet said anything about how we might choose such a surrogate function, \( q(x; x') \). In fact, the choice of surrogate function will depend on the specific problem to be solved, but for now we will consider the general case.

Now in order to make the intuition of this figure a bit more formal, imagine that we have an algorithm for globally minimizing the function \( q(x; x') \) as a function of \( x \) from an initial state given by \( x' = x^{(k)} \). Then using the initial state \( x^{(k)} \), we can compute an updated state, \( x^{(k+1)} \), using the following iteration known as the **majorization minimization algorithm**.

\[
x^{(k+1)} = \text{arg } \min_{x \in \mathbb{R}^N} \left\{ q(x; x^{(k)}) \right\}
\]

The following sequence of inequalities then guarantees that the value of \( f(x) \) is reduced or at least stays the same with each iteration.

\[
f(x^{(k+1)}) \leq q(x^{(k+1)}; x^{(k)}) \leq q(x^{(k)}; x^{(k)}) = f(x^{(k)})
\]

Notice that the first inequality results from condition (8.2); the second inequality results from the minimization of equation (8.3); and the third equality results from condition (8.1). Therefore, this proves that the majorization minimization procedure is guaranteed to produce a monotone decreasing sequence of values of the true cost function. If the cost function is strictly positive (or just bounded below), then (8.4) also guarantees that the resulting sequence of cost function values, \( f(x^{(k)}) \), must converge.\(^2\) Again, Figure 8.1 illustrates the intuition behind the inequalities of equation (8.4). Since \( q(x; x^{(k)}) \) upper bounds \( f(x) \), reducing the value of \( q(x; x^{(k)}) \) must guarantee that \( f(x) \) is also reduced.

### 8.2 Formal Definition and Properties

Now that we understand the intuition behind surrogate functions, we can make the framework more precise with the introduction of some formal definitions and properties. Now in practice, we can actually relax the conditions

\(^2\) A common misconception is to assume that the sequence of actual values, \( x^{(k)} \), must also be convergent. However, while this is typically true for practical implementations of the majorization minimization, it is not guaranteed by the conditions of equations (8.1), (8.2), and (8.3).
we gave for the surrogate function and still retain the desired outcome. In particular, let \( c(x') \) be any function of the initial state \( x' \). If we define the function
\[
Q(x; x') = q(x; x') + c(x') ,
\]
then minimization of the function \( Q(x; x') \) results in the same update of equation (8.3); so it can also be used to substitute for \( f(x) \) in the optimization. In particular, if we are given the function \( Q(x; x') \), we can easily compute the function \( q(x; x') \) by appropriately shifting the \( Q \) function.

\[
q(x; x') = Q(x; x') - Q(x'; x') + f(x') \tag{8.5}
\]

Notice that with this shift, we ensure the condition of equation (8.1) that \( f(x') = q(x'; x') \).

Substituting the expression of equation (8.5) into the condition of equation (8.2) leads to our definition for a valid surrogate function.

**Definition 10. Surrogate Function**

Let \( f(x) \) be a function on \( A \subseteq \mathbb{R}^N \). Then we say that \( Q(x; x') \) is a surrogate function for the minimization of \( f(x) \) if for all \( x, x' \in A \) the following condition holds.

\[
f(x) \leq Q(x; x') - Q(x'; x') + f(x') \tag{8.6}
\]

Similarly, we say that \( Q(x; x') \) is a surrogate function for the maximization of \( f(x) \) if for all \( x, x' \in A \)

\[
f(x) \geq Q(x; x') - Q(x'; x') + f(x') \tag{8.7}
\]

Here we will treat the problem of minimization, but surrogate functions are equally applicable to maximization, and all the equivalent results can be obtained by simply negating the functions being maximized.

Armed with this definition, surrogate function minimization is then performed using the very simple majorization minimization algorithm of Figure 8.2. Basically, each iteration of the algorithm minimizes the value of the surrogate function, and each new surrogate function uses the value of \( x^{(k-1)} \) determined from the previous iteration.

The simple assumptions we have made about the surrogate function will ensure some very powerful properties of the associated minimization algo-
8.2 Formal Definition and Properties

Below we list five such properties. The first property results from a simple rearrangement of the terms of equation (8.6).

**Property 8.1. Monotonicity property of majorization** - Let \( Q(x; x') \) be a surrogate function for minimization of \( f(x) \) on \( A \subset \mathbb{R}^N \). Then for all \( x, x' \in A \),

\[
\{ Q(x; x') < Q(x'; x') \} \Rightarrow \{ f(x) < f(x') \} ,
\]

(8.8)

and if \( Q(x; x') = Q(x'; x') \), then \( f(x) \leq f(x') \).

The next property states that the gradients of the \( Q \) and \( f \) functions must be equal at \( x = x' \). This results from the fact that the functions must be tangent; however, it only holds when \( x' \in \text{int}(A) \).

**Property 8.2. Gradient property of surrogate function** - Let \( Q(x; x') \) be a surrogate function for minimization of \( f(x) \) on \( A \subset \mathbb{R}^N \). Further, assume that both \( Q \) and \( f \) are continuously differentiable functions of \( x \). Then for all \( x' \in \text{int}(A) \),

\[
\nabla f(x)|_{x=x'} = \nabla Q(x; x')|_{x=x'} .
\]

(8.9)

The next property states that if \( x^* \) is a fixed point of the majorization update algorithm, then it must also be a point at which the gradient of the cost function is zero.

**Property 8.3. Fixed point property of majorization** - Let \( Q(x; x') \) be a surrogate function for minimization of \( f(x) \) on \( A \subset \mathbb{R}^N \) where both \( Q \) and \( f \) are continuously differentiable functions of \( x \). Then for all \( x^* \in \text{int}(A) \),

\[
\left\{ Q(x^*; x^*) = \inf_{x \in A} Q(x; x^*) \right\} \Rightarrow \{ \nabla f(x^*) = 0 \} .
\]

(8.10)

This means that for convex differentiable functions on \( \mathbb{R}^N \), any fixed point of the majorization minimization algorithm must correspond to a local minimum of the actual function, \( f \). Furthermore, if \( f \) is strictly convex, then any such local minimum must also be a global minimum.

The fourth property ensures finite additivity of surrogate functions.
Property 8.4. Additivity property of surrogate functions - For \( k = 1, \cdots, K \), let \( Q_k(x; x') \) be a surrogate function of \( f_k(x) \) on \( A \subset \mathbb{R}^N \). Furthermore, for all \( x, x' \in A \) define

\[
    f(x) \triangleq \sum_{k=1}^{K} f_k(x) \\
    Q(x; x') \triangleq \sum_{k=1}^{K} Q_k(x; x') .
\]

Then \( Q(x; x') \) is a surrogate function for \( f(x) \) on \( A \).

This property will be extremely useful because it ensures that we can design surrogate functions for individual terms of a cost function, and then sum them all together to form a complete surrogate function for our problem.

Finally, composition of the true and surrogate functions with a third function still results in a valid surrogate function.

Property 8.5. Composition property of surrogate functions - Let \( Q(x; x') \) be a surrogate function for minimization of \( f(x) \) on \( A \subset \mathbb{R}^N \); and let \( g : A \to A \). Then define

\[
    \tilde{f}(x) \triangleq f(g(x)) \\
    \tilde{Q}(x; x') \triangleq Q(g(x); g(x')) .
\]

Then \( \tilde{Q}(x; x') \) is a surrogate function for \( \tilde{f}(x) \).

8.3 Minimizing the MAP Cost Function

Let us next see how majorization techniques can be applied to the MAP cost function of equation (7.21). For this case, the function to be minimized is given by

\[
    f(x) = \frac{1}{2} \| y - Ax \|_A^2 + \sum_{\{s,r\} \in P} b_{s,r} \rho(x_s - x_r) .
\]

(8.11)

Since the data term is quadratic, it already can be handled with the optimization techniques described in Chapter 5. Therefore, we need to find a quadratic surrogate function that can be used to replace the prior term of
8.3 Minimizing the MAP Cost Function

Majorization Minimization Algorithm:

Initialize $x'$

For $k = 1$ to $K$

\[ x' \leftarrow \arg \min_{x \in \mathcal{A}} Q(x; x') \]

\[ x^* = x' \]

Figure 8.2: This figure provides a pseudocode specification of the majorization minimization algorithm. In each iteration, the true function, $f(x)$, is approximated by a surrogate function, $Q(x; x')$, in order to compute the minimum. The properties of the surrogate function ensure that this produces a monotonically decreasing sequence of values $f(x')$.

$f(x)$. We can do this by first finding a surrogate function for the potential function, $\rho(\Delta)$, and then applying the results of Properties 8.4 and 8.5. More specifically, if we can determine a surrogate potential function, $\rho(\Delta; \Delta')$, which is a quadratic function of $\Delta$, then we may use $\rho(\Delta; \Delta')$ to construct a surrogate MAP cost function with the following form

\[
Q(x; x') = \frac{1}{2} ||y - Ax||^2_{\Lambda} + \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} \rho(x_s - x_r; x'_s - x'_r) .
\] (8.12)

Once this is done, $Q(x; x')$ is then a quadratic function of $x$, and the optimization problem has the same form as considered in Chapter 5. Repeated optimization of the $Q$ function, as specified by the majorization minimization algorithm of Figure 8.2, will then also minimize the true MAP cost function $f(x)$.

So our objective is to find a surrogate function, $\rho(\Delta; \Delta')$, for the potential function $\rho(\Delta)$. There are a number of strategies for doing this, and the best choice tends to depend on the specific structure of the potential function being minimized.

We will first use the **maximum curvature** method to construct a surrogate function. To do this, assume a surrogate function of the form,

\[
\rho(\Delta; \Delta') = a_1 \Delta + \frac{a_2}{2} (\Delta - \Delta')^2 .
\]

By simply matching the gradients of $\rho(\Delta)$ and $\rho(\Delta; \Delta')$ at $\Delta = \Delta'$ as described in Property 8.2, we can find the value of the coefficient $a_1$.

\[
a_1 = \rho'(\Delta')
\]
Now in order to ensure that the surrogate function is an upper bound to the true function, we will select the value of \( a_2 \) to be the maximum of the potential function’s second derivative.

\[
a_2 = c_{\text{max}} \triangleq \max_{\Delta \in \mathbb{R}} \rho''(\Delta) \tag{8.13}
\]

Notice that for a given potential function, \( c_{\text{max}} \) does not change; so only \( a_1 \) must be recomputed each time \( \Delta' \) changes. Putting this all together, we get the following maximum curvature surrogate function for \( \rho(\Delta) \).

\[
\rho(\Delta; \Delta') = \rho'(\Delta') \Delta + \frac{c_{\text{max}}}{2} (\Delta - \Delta')^2 \tag{8.14}
\]

Plugging this surrogate function into the expression of equation (8.12) results in a complete expression for the MAP surrogate cost function using the maximum curvature method.

\[
Q(x; x') = \frac{1}{2} ||y - Ax||_A^2 + \sum_{\{s, r\} \in \mathcal{P}} b_{s, r} \rho'(x'_s - x'_r) (x_s - x_r) \tag{8.15}
+ \sum_{\{s, r\} \in \mathcal{P}} \frac{b_{s, r} c_{\text{max}}}{2} (x_s - x'_s - (x_r - x'_r))^2
\]

A disadvantage of the maximum curvature method is that it tends to be too conservative in the choice of the curvature \( a_2 \). This is particularly true if the second derivative of the potential function is large near zero. In fact, if the second derivative is infinite, then there is no maximum, and the method cannot be used.

The **symmetric bound** method can be used in place of the maximum curvature method. The advantage of this method is that it usually results in a much less conservative upper bound; so it tends to yield optimization algorithms that converge much faster than those of the maximum curvature method. The symmetric bound method works for all the potential functions in Section 6.3 and is illustrated in Figure 8.3. In this strategy, each potential function from equation (7.6) is upper bounded by a surrogate function that is a symmetric and quadratic function of \( \Delta \). Using this assumption, the surrogate function has the form

\[
\rho(\Delta; \Delta') = \frac{a_2}{2} \Delta^2 ,
\]
surrogate function
\( \rho(\Delta; \Delta') + \text{const} \)

function to be minimized
\( \rho(\Delta) \)

Figure 8.3: Figure showing the intuition behind the symmetric bound surrogate function of equation (8.17). For some convex functions, the value of the second derivative can be chosen so that the function touches and is tangent to the potential function at two symmetric points about 0. This tends to produce a much tighter surrogate function.

where \( a_2 \) is a function of \( \Delta' \).

In order to determine \( a_2 \), we again match the gradients of \( \rho(\Delta) \) and \( \rho(\Delta; \Delta') \) at \( \Delta = \Delta' \) to yield

\[
a_2 = \frac{\rho'(\Delta')}{\Delta'},
\]

which results in the following symmetric bound surrogate function,

\[
\rho(\Delta; \Delta') = \begin{cases} 
\frac{\rho'(\Delta')}{2\Delta'} \Delta^2 & \text{if } \Delta' \neq 0 \\
\frac{\rho''(0)}{2} \Delta^2 & \text{if } \Delta' = 0
\end{cases},
\]

where we use the limiting value of \( a_2 = \rho''(0) \) when \( \Delta' = 0 \).

Of course, the question remains as to whether this function of equation (8.17) will achieve the defining constraint of equation (8.6). It can be shown that in fact the constraint is met for any potential function with the property that its second derivative decreases as \(|\Delta|\) increases. This is also equivalent to the condition that the influence function \( \rho'(\Delta) \) is concave for \( \Delta > 0 \); or equivalently, that the influence function is convex for \( \Delta < 0 \). In
practice, this is a common property for practical potential functions since it means that the influence of pixels will decrease as the difference between pixels increases.

Table 8.1 lists out the analytical expressions for the terms of equation (8.17) for each of the potential functions we have discussed. Notice that some of the expressions for $\rho''(0)/2$ are unbounded. For example, this is the case for the total variation potential function because its second derivative becomes unbounded at zero. This represents a real practical problem when using surrogate functions. In fact, a major advantage of the Huber and QGGMRF potentials is that they have bounded second derivatives at zero; so both these potential functions are compatible with surrogate optimization methods. In Chapter 9 we will address this problem by introducing new algorithms that can be used when computing the MAP estimate with potential functions such as the total variation prior.

Substituting the symmetric bound surrogate function of (8.17) into equa-

<table>
<thead>
<tr>
<th>Name</th>
<th>$\rho(\Delta)$</th>
<th>$\frac{\rho'(\Delta')}{2\Delta'}$</th>
<th>$\frac{\rho''(0)}{2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quadratic</td>
<td>$\frac{\Delta^2}{2\sigma_x^2}$</td>
<td>$\frac{1}{2\sigma_x^2}$</td>
<td>$\frac{1}{2\sigma_x^2}$</td>
</tr>
<tr>
<td>Total Variation</td>
<td>$\frac{</td>
<td>\Delta</td>
<td>}{\sigma_x}$</td>
</tr>
<tr>
<td>Huber</td>
<td>$\begin{cases} \frac{\Delta^2}{2\sigma_x^2} &amp; \text{for }</td>
<td>\Delta</td>
<td>&lt; T\sigma_x \ T\frac{</td>
</tr>
<tr>
<td>log cosh</td>
<td>$T^2 \log \cosh \left( \frac{\Delta}{T\sigma_x} \right)$</td>
<td>$\frac{T \tanh \left( \frac{\Delta'}{T\sigma_x} \right)}{\sigma_x \Delta'}$</td>
<td>$\frac{1}{2\sigma_x^2}$</td>
</tr>
<tr>
<td>GGMRF</td>
<td>$\frac{</td>
<td>\Delta</td>
<td>^p}{p\sigma_x^p}$</td>
</tr>
<tr>
<td>QGGMRF ($p \leq q$)</td>
<td>$\frac{</td>
<td>\Delta</td>
<td>^p}{p\sigma_x^p} \left( \frac{\frac{</td>
</tr>
</tbody>
</table>
tion (8.12) results in the following MAP surrogate cost function,

\[
Q(x; x') = \frac{1}{2}||y - Ax||_A^2 + \sum_{\{s, r\} \in P} \frac{b_{s,r} \rho'(x'_s - x'_r)}{2(x'_s - x'_r)} (x_s - x_r)^2 ,
\]  

(8.18)

where for simplicity we assume that \(x'_s - x'_r \neq 0\). Below is an example of how this method can be applied for a particular choice of potential function.

<table>
<thead>
<tr>
<th>Name</th>
<th>Surrogate Function Coefficient ( \tilde{b}_{s,r} ) of Equation (8.19)</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>( \tilde{b}<em>{s,r} \leftarrow b</em>{s,r} \frac{\rho'(x'_s - x'_r)}{2(x'_s - x'_r)} )</td>
</tr>
<tr>
<td>Quadratic</td>
<td>( \tilde{b}<em>{s,r} \leftarrow b</em>{s,r} \frac{1}{2\sigma_x^2} )</td>
</tr>
<tr>
<td>Total Variation</td>
<td>( \tilde{b}<em>{s,r} \leftarrow b</em>{s,r} \frac{1}{2\sigma_x</td>
</tr>
<tr>
<td>Huber</td>
<td>( \tilde{b}<em>{s,r} \leftarrow b</em>{s,r} \frac{T \tanh \left( \frac{x'_s - x'_r}{\sigma_x} \right)}{2\sigma_x^2(x'_s - x'_r)} )</td>
</tr>
<tr>
<td>log cosh</td>
<td>( \tilde{b}<em>{s,r} \leftarrow b</em>{s,r} \frac{T \tanh \left( \frac{x'_s - x'_r}{\sigma_x} \right)}{2\sigma_x^2(x'_s - x'_r)} )</td>
</tr>
<tr>
<td>GGMRF</td>
<td>( \tilde{b}<em>{s,r} \leftarrow b</em>{s,r} \frac{</td>
</tr>
<tr>
<td>QGGMRF</td>
<td>( \tilde{b}<em>{s,r} \leftarrow b</em>{s,r} \frac{</td>
</tr>
</tbody>
</table>

Table 8.2: Analytical expressions for the coefficient \( \tilde{b}_{s,r} \) required for computing the surrogate function using the symmetric bound method. Each expression corresponds to a potential function given in Table 6.1 of Section 6.4.

An alternative way to think about the surrogate function approach is that with each iteration, we recompute the local coefficients, \( b_{s,r} \). Using this point of view, we can express the surrogate MAP cost function as

\[
\tilde{b}_{s,r} \leftarrow b_{s,r} \frac{\rho'(x'_s - x'_r)}{2(x'_s - x'_r)} ,
\]  

(8.19)

\[
Q(x; x') = \frac{1}{2}||y - Ax||_A^2 + \sum_{\{s, r\} \in P} \tilde{b}_{s,r} (x_s - x_r)^2 ,
\]  

(8.20)
where the coefficients, $\tilde{b}_{s,r}$, are the modified coefficients that are updated with each iteration of an optimization algorithm.

So in practice, in order to implement a particular potential function one simply computes the values of $\tilde{b}_{s,r}$ corresponding to their desired choice of potential function. Table 8.2 lists out the specific formulas for the coefficients, $\tilde{b}_{s,r}$, for each of the potential functions we have considered in Table 6.1. While the formulas might seem a bit messy, the approach is straightforward. One simply calculates the value of $\tilde{b}_{s,r}$ corresponding to the desired potential function, and then minimizes the quadratic cost function of equation (8.20) with respect to $x$.

**Example 8.1.** In this example, we find the symmetric bound surrogate function for the Huber potential listed in Table 6.1 of Section 6.4. In this case, the potential function is given by

$$\rho(\Delta) = \begin{cases} \frac{\Delta^2}{2\sigma_x^2} & \text{for } |\Delta| < T\sigma_x \\ T\epsilon\left|\frac{\Delta}{\sigma_x}\right| - \frac{T^2}{2} & \text{for } |\Delta| \geq T\sigma_x \end{cases},$$

and its associated influence function is given by

$$\rho'(\Delta) = \frac{1}{\sigma_x^2} \text{clip}\{\Delta, [-T\sigma_x, T\sigma_x]\},$$

where the function $\text{clip}\{x, [a, b]\}$ clips the value $x$ to the interval $[a, b]$. For the symmetric bound, the surrogate function of equation (8.17) is given by

$$\rho(\Delta; \Delta') = \frac{\text{clip}\{\Delta', [-T\sigma_x, T\sigma_x]\}}{2\sigma_x^2\Delta'} \Delta'^2. \quad (8.21)$$

Notice that the result of equation (8.21) corresponds to the entry labeled “Huber” in Table 8.2. So in practice, we could have skipped this analysis, and referred directly to the Table to find the update for the $\tilde{b}_{s,r}$ terms given by

$$\tilde{b}_{s,r} \left\{ \begin{array}{ll} b_{s,r} \frac{\text{clip}\{x'_s - x'_r, [-T\sigma_x, T\sigma_x]\}}{2\sigma_x^2(x'_s - x'_r)} & \text{for } x'_s \neq x'_r \\ b_{s,r} \frac{2\sigma_x^2}{2\sigma_x^2} & \text{for } x'_s = x'_r \end{array} \right.$$
where we explicitly add the case when \( x'_s = x'_r \). From this, the symmetric bound MAP surrogate cost function is given by

\[
Q(x; x') = \frac{1}{2} ||y - Ax||^2_A + \sum_{\{s,r\} \in P} \tilde{b}_{s,r} (x_s - x_r)^2.
\] (8.22)

The MAP estimate can then be computed though repeated minimization of (8.22). Since the function is now a quadratic function of \( x \), a wide variety of optimization algorithms can be used for this purpose.

### 8.4 Application to Line Search

In order to better understand how surrogate functions can be used, let us next consider its application to the important problem of line-search optimization discussed in Section 7.2. In this case, our objective is to solve the 1-D minimization problem

\[
\alpha^* = \arg \min_{\alpha \in [a, b]} f(x + \alpha d),
\]

where \( x \in \mathbb{R}^N \) is the current value of the solution, \( d \in \mathbb{R}^N \) is the update direction, \( \alpha \) is the unknown update step size, and \([a, b]\) is a search interval that includes 0.

Using a surrogate function in place of \( f(x) \), we compute the modified line search

\[
\alpha^* = \arg \min_{\alpha \in [a, b]} Q(x + \alpha d; x),
\]

where we assume that \( \alpha \in [a, b] \). Now since the surrogate MAP cost function of equation (8.22) is a quadratic function of \( x \), we can use the same trick and express the line search function as a quadratic function of \( \alpha \),

\[
Q(x + \alpha d; x) = c + \theta_1 \alpha + \frac{1}{2} \theta_2 \alpha^2,
\]

where it can be shown that (See problem 4)

\[
\theta_1 = -(y - Ax)^t \Lambda A d + \sum_{\{s,r\} \in P} 2\tilde{b}_{s,r} (x_s - x_r) (d_s - d_r)
\] (8.23)

\[
\theta_2 = d^t A^t \Lambda A d + \sum_{\{s,r\} \in P} 2\tilde{b}_{s,r} (d_s - d_r)^2.
\] (8.24)
General MAP Optimization using Majorization of Prior:

Initialize \( x \)

For \( K \) iterations {
\[
\{d, a, b\} \leftarrow \text{SelectSearchDirectionAndInterval}(x)
\]
\[
\bar{b}_{s,r} \leftarrow \frac{b_{s,r} \rho'(x_s - x_r)}{2(x_s - x_r)}
\]
\[
\theta_1 \leftarrow -(y - Ax)'^\top \Lambda Ad + \sum_{\{s,r\} \in P} 2\bar{b}_{s,r} (x_s - x_r) (d_s - d_r)
\]
\[
\theta_2 \leftarrow d'^\top A'^\top \Lambda Ad + \sum_{\{s,r\} \in P} 2\bar{b}_{s,r} (d_s - d_r)^2
\]
\[
\alpha^* \leftarrow \text{clip}\left\{\frac{-\theta_1}{\theta_2}, [a, b]\right\}
\]
\[
x \leftarrow x + d\alpha^*
\]

Figure 8.4: Pseudocode for the implementation of a general MAP optimization procedure that uses majorization of the prior term. Each line search is approximately solved by replacing the true prior term with a surrogate function.

In this case, the line search solution has the very simple form of

\[
\alpha^* = \text{clip}\left\{\frac{-\theta_1}{\theta_2}, [a, b]\right\}.
\]

Figure 8.4 provides a pseudocode specification of a generic optimization algorithm which uses this majorization approach in the line search step. The search direction may vary based on the specific choice of optimization algorithm, but the computation of the line search always follows the same procedure. Notice that the resulting value of \( \alpha^* \) guarantees that

\[
f(x + \alpha^* d) - f(x) \leq Q(x + \alpha^* d; x) - Q(x; x) \leq 0,
\]

so we know that the true cost function decreases with each iteration of this algorithm. The advantage of this modified line search is that the 1-D optimization can be computed very efficiently in one step. This is in contrast to the many iterations that may be required of the half-interval search of Section 7.2.

Example 8.2. In this example, we derive the explicit expressions for ICD updates using a symmetric surrogate function. For the ICD algorithm, the
8.4 Application to Line Search

ICD Algorithm using Majorization of Prior:

Initialize \( e \leftarrow y - Ax \)

For \( K \) iterations {
    For each pixel \( s \in S \) {
        \( \tilde{b}_{s,r} \leftarrow \frac{b_{s,r} \rho'(x_s - x_r)}{2(x_s - x_r)} \)
        \( \theta_1 \leftarrow -e^t \Lambda A_{*,s} + \sum_{r \in \partial s} 2\tilde{b}_{s,r} (x_s - x_r) \)
        \( \theta_2 \leftarrow A_{*,s} \Lambda A_{*,s} + \sum_{r \in \partial s} 2\tilde{b}_{s,r} \)
        \( \alpha^* \leftarrow \text{clip}\left\{ \frac{-\theta_1}{\theta_2}, [-x_s, \infty) \right\} \)
        \( x_s \leftarrow x_s + \alpha^* \)
        \( e \leftarrow e - A_{*,s} \alpha^* \)
    }
}

Figure 8.5: Pseudocode for fast implementation of the ICD algorithm with majorization of the prior term. The speedup depends on the use of a state variable to store the error, \( e = y - Ax \). Each line search is approximately solved by replacing the true prior term with a surrogate function.

update direction is given by \( d = \varepsilon_s \) where \( \varepsilon_s \) is a vector that is 1 for element \( s \), and 0 otherwise. So using equations (8.23) and (8.24), we can derive a simplified expression for the first and second derivatives of the ICD line search.

\[
\theta_1 = -e^t \Lambda A_{*,s} + \sum_{r \in \partial s} 2\tilde{b}_{s,r} (x_s - x_r) \tag{8.25}
\]

\[
\theta_2 = A_{*,s} \Lambda A_{*,s} + \sum_{r \in \partial s} 2\tilde{b}_{s,r} \tag{8.26}
\]

where \( e = y - Ax \) is the error in the measurement domain. If \( A \) is a sparse matrix, then keeping this state vector, \( e \), can dramatically reduce the computation required for an ICD update.

Again, Figure 8.5 provides a pseudocode specification of this complete algorithm. Notice that, while the surrogate function approach does not compute the exact local minimum of the line search problem, it does guarantee monotone convergence, and in practice it results in a fast and computation-
ally efficient algorithm.
8.5 Chapter Problems

1. Let \( q(x; x') \) be a surrogate function for the minimization of \( f(x) \) so that \( \forall x', x \in \mathbb{R}^N \),

\[
  f(x') = q(x'; x')
  f(x) \leq q(x; x').
\]

Prove that \( \forall x^{(k)} \in \mathbb{R}^N \), when \( x^{(k+1)} = \arg\min_{x \in \mathbb{R}^N} \{q(x; x^{(k)})\} \), then \( f(x^{(k+1)}) \leq f(x^{(k)}) \).

2. Let \( Q(x; x') \) be a surrogate function for the minimization of \( f(x) \) such that both \( Q \) and \( f \) are continuously differentiable functions of \( x \).

a) Prove the result of Property 8.2 that \( \forall x' \in \mathbb{R}^N \),

\[
  \nabla f(x)|_{x=x'} = \nabla Q(x; x')|_{x=x'}.
\]

b) Prove the result of Property 8.3, that \( \forall x^* \in \mathbb{R}^N \),

\[
  \left\{ Q(x^*; x^*) = \inf_{x \in \mathbb{R}^N} Q(x; x^*) \right\} \Rightarrow \{ \nabla f(x^*) = 0 \}.
\]

3. Prove that if the potential function \( \rho(\Delta) \) has the property that its associated influence function \( \rho'(\Delta) \) is concave for \( \Delta > 0 \) (or equivalently, convex for \( \Delta < 0 \)), then the symmetric bound surrogate function of equation (8.17) produces an upper bound to the true function.

4. Provide a counter example to the proof of problem 3 when the influence function is not concave for \( \Delta > 0 \). (Hint: Try the function \( \rho(\Delta) = |\Delta|^p \) for \( p > 2 \).)

5. Consider the function

\[
  f(x) = |x - x_r|^{1.1},
\]

for \( x \in \mathbb{R} \).

a) Sketch a plot of \( f(x) \) when \( x_r = 1 \).

b) Sketch a good surrogate function, \( f(x; x') \), for \( x_r = 1 \) and \( x' = 2 \).

c) Determine a general expression for the surrogate function \( f(x; x') \) that works for any value of \( x_r \) and \( x' \).
d) Assuming the objective is to minimize the expression

\[ f(x) = \sum_{r \in \partial s} |x - x_r|^{1.1}, \]

for \( x \in \mathbb{R} \), specify an iterative algorithm in terms of the surrogate function \( f(x; x') \) that will converge to the global minimum of the function.

6. Let \( \rho(x) \) for \( x \in \mathbb{R} \) be the Huber function for \( T = 1 \). The objective of this problem is to determine surrogate functions, \( Q(x; x') \), for \( \rho(x) \).

a) Use the maximum curvature method to determine \( Q(x; x') \).

b) From \( Q(x; x') \), determine the function \( q(x; x') \), so that \( q(x; x) = f(x) \).

c) Plot the functions \( \rho(x) \) and \( q(x; x') \) for \( x' = 4 \) and indicate the location of the minimum

\[ x'' = \arg \min_{x \in \mathbb{R}} q(x; 4) \]

d) Use the symmetric bound method to determine \( Q(x; x') \).

e) From \( Q(x; x') \), determine the function \( q(x; x') \), so that \( q(x; x) = f(x) \).

f) Plot the functions \( \rho(x) \) and \( q(x; x') \) for \( x' = 4 \) and indicate the location of the minimum

\[ x'' = \arg \min_{x \in \mathbb{R}} q(x; 4) \]

g) What are the advantages and disadvantages of maximum curvature and symmetric methods?
Chapter 9

Constrained Optimization

In recent years, there has been a growing awareness that constrained optimization can be of great importance not only as a way in which to enforce physical constraints, but also has a clever strategy for making large and difficult optimization problems numerically more tractable. This is particularly interesting since it runs against the natural intuition that unconstrained optimization is easier than constrained optimization.

The goal of this chapter is to introduce the essential machinery of constrained optimization and then illustrate the power of these methods with some canonical examples. The chapter starts by first motivating the problem, and then covering the basic theory of constrained optimization including the theory and methods of the augmented Lagrangian. We show how the augmented Lagrangian provides a practical framework for solving large numerical optimization problems with constraints. With these tools in hand, we then introduce variable splitting, the proximal map, and the ADMM algorithm as methods for decomposing large and difficult MAP optimization problems into smaller more manageable ones. Finally, we illustrate the power of these approaches by formulating solutions to some very important optimization problems including total variation regularization and MAP optimization with positivity constraints.

9.1 Motivation and Intuition

The methods of constrained optimization have grown to be very important in model-based imaging. For example, positivity constraints are very common in
imaging problems since the unknown, $X$, often represents quantities related to energy or absorption, which must be positive for physical reasons. In this case, the need for constrained optimization is obvious since the MAP estimate must be computed using the convex constraint that $X \in \mathbb{R}^{+N}$.

However, in recent years, it has also become clear that constrained optimization techniques can be extremely useful for solving large unconstrained MAP optimization problems. In order to illustrate this counter-intuitive fact, we start with a familiar problem which requires the unconstrained optimization of two separate terms.

\[
\hat{x} = \arg \min_{x \in \mathbb{R}^N} \{f(x) + g(x)\} \tag{9.1}
\]

In fact, this problem has the form of a typical MAP optimization problem where $f(x)$ is the negative log likelihood and $g(x)$ is the negative log prior probability.

First notice that the unconstrained optimization problem of equation (9.1) can also be used to enforce positivity constraints by setting $g(x) = \infty$ for $x \notin \mathbb{R}^{+N}$. So for example, if we choose

\[
\tilde{g}(x) = \begin{cases} 
0 & \text{if } x \in \mathbb{R}^{+N} \\
\infty & \text{if } x \notin \mathbb{R}^{+N}
\end{cases}
\]

Then in this case, we have that

\[\arg \min_x \{f(x) + \tilde{g}(x)\} = \arg \min_{x \in \mathbb{R}^{+N}} f(x). \tag{9.2}\]

Intuitively, adding $\tilde{g}$ enforces the positivity by making the cost function infinite when $x$ moves outside the constraint. We say that $\tilde{g}$ is an extended function because it takes values on $\mathbb{R} \cup \{\infty\}$. Perhaps surprisingly, $\tilde{g} : \mathbb{R}^N \to \mathbb{R} \cup \{\infty\}$ is also a convex function if we apply the standard definition of convexity using the very reasonable convention that $\infty + a = \infty$. Moreover, $\tilde{g}$ is also a proper and closed convex function using the definitions given in Appendix A.

Now we will do something very counter-intuitive. We will take a perfectly good unconstrained optimization problem with a single independent variable, $x$, and we will “split” the variable into two variables, $x$ and $v$, with the constraint that $x = v$. So then the new, but fully equivalent, problem is
given by
\[
(\hat{x}, \hat{v}) = \arg\min_{(x,v)} \left\{ f(x) + g(v) \right\}.
\] (9.3)

Well, that certainly seems like a silly thing to do. Of course, given the constraint it must be that \( \hat{x} = \hat{v} \) and therefore that the solution to equations (9.1) and (9.3) must be the same. But why would anyone want to do such a thing?

Well rather than answering that question now, we will continue to forge ahead and solve this constrained optimization problem so we can see where it leads us. In order to enforce the constraint, we will add an additional term to the cost function that penalizes large differences between \( x \) and \( v \). This then results in the unconstrained optimization problem given by
\[
(\hat{x}, \hat{v}) = \arg\min_{(x,v)} \left\{ f(x) + g(v) + a \frac{1}{2} ||x - v||^2 \right\},
\] (9.4)

where \( a \) is a constant that controls the gain of the penalty term.

Now as the value of \( a \) is increased, the constraint of \( x = v \) is more aggressively enforce. So by making \( a \) very large, we should be able to force \( \hat{x} \approx \hat{v} \). However, no matter how large we make \( a \), it will likely never be the case that \( \hat{x} = \hat{v} \); so the solution to equation (9.4) will not be exactly the same as the original problem of equations (9.1). In order to fix this problem, we will need to augment the problem with an additional term that can “push” the solution towards a result that exactly meets the constraint \( \hat{x} = \hat{v} \). This can be done by using the following modified unconstrained optimization problem
\[
(\hat{x}, \hat{v}) = \arg\min_{(x,v)} \left\{ f(x) + g(v) + a \frac{1}{2} ||x - v + u||^2 \right\},
\] (9.5)

where \( u \) must be chosen to achieve the desired constraint that \( \hat{x} = \hat{v} \).

In fact, in Section 9.3, we will see that the expression being minimized in equation (9.5) is known as the augmented Lagrangian for our constrained optimization problem and the term \( u \) serves a role equivalent to that of a Lagrange multiplier. The beauty of this approach is that the correct value of \( u \) can be determined using the following simple augmented Lagrangian algorithm.
Split-Variable Augmented Lagrangian Algorithm

Initialize $u \leftarrow 0$ and $\hat{v} \leftarrow 0$

For $K$ iterations {

$$(\hat{x}, \hat{v}) \leftarrow \arg \min_{(x,v)} \left\{ f(x) + g(v) + \frac{a}{2} \|x - v + u\|^2 \right\}$$

$$u \leftarrow u + (\hat{x} - \hat{v})$$

}

So with each iteration, the value of $u$ is increased (or decreased) by the difference $\delta = \hat{x} - \hat{v}$. Intuitively, if $\delta$ is positive, then $u \leftarrow u + \delta$ increases the value of $u$, which then “pushes” $\delta$ down until of $\hat{x} = \hat{v}$. Alternatively, if $\delta$ is negative, then $u \leftarrow u + \delta$ decreases the value of $u$, which then “pushes” $\delta$ up until of $\hat{x} = \hat{v}$. Either way, this iterative process drives $u$ to a value which causes the constraint of $\hat{x} = \hat{v}$ to be met. Later in Section 9.2, we will introduce the basic machinery of convex optimization including the duality principle, and in Section 9.3 we will use this machinery to prove that this augmented Lagrangian algorithm converges to the correct value of $u$.

With the problem in this form, any engineer worth their salt should have a strong desire to approximately perform the optimization of $(x, v)$ in two steps using alternating minimization of $x$ and $v$. Of course, it won’t produce the same answer, but it should be close, and it seems like it would be much easier to do. This alternating optimization approach is shown below and known as the alternating directions method of multipliers or ADMM algorithm.

Split-Variable ADMM Algorithm

Initialize $u \leftarrow 0$ and $\hat{v} \leftarrow 0$

For $K$ iterations {

$$\hat{x} \leftarrow \arg \min_{x} \left\{ f(x) + \frac{a}{2} \|x - \hat{v} + u\|^2 \right\}$$

$$\hat{v} \leftarrow \arg \min_{v} \left\{ g(v) + \frac{a}{2} \|\hat{x} - v + u\|^2 \right\}$$

$$u \leftarrow u + (\hat{x} - \hat{v})$$

}

As it turns out, the ADMM algorithm is also convergent for convex opti-
9.1 Motivation and Intuition

Now we return to the original question of, why would we choose to solve the problem of equation (9.1) using this split-variable augmented Lagrangian algorithm rather than direct optimization? The reason is that for many important optimization problems the two functions, $f(x)$ and $g(x)$, are so different that a single optimization approach is not appropriate for both. Since the split-variable ADMM algorithm allows the two expressions to be minimized separately, each term may be minimized with a different algorithm that is best suited to its structure. Intuitively, the ADMM approach allows the two variables $x$ and $v$ to move along separate paths and then converge together to a single coherent solution.

Perhaps this can be best illustrated by again considering the problem of optimization with a positivity constraint as in equation (9.2). In this case, the optimization over $v$ can be computed in closed form and is given by

$$
\hat{v} = \arg \min_v \left\{ \tilde{g}(v) + \frac{a}{2} ||\hat{x} - v + u||^2 \right\} \\
= \arg \min_{v \in \mathbb{R}^+ N} \left\{ \frac{a}{2} ||\hat{x} + u - v||^2 \right\} \\
= \text{clip}\{\hat{x} + u, \mathbb{R}^+ N\} ,
$$

where we define the function $\text{clip}\{x, \mathcal{A}\}$ that projects or “clips” any vector $x$ onto the convex set $\mathcal{A}$. (See Appendix A) For the special case of non-negative vectors, we have that $\mathcal{A} = \mathbb{R}^+ N$ and for each component

$$
[\text{clip}\{x, \mathbb{R}^+ N\}]_i = \text{clip}\{x_i, [0, \infty)\} ,
$$

so each component is clipped to be non-negative.

Applying the split-variable ADMM algorithm to this problem, we then get the following algorithm for enforcing positivity in the minimization of any convex function $f(x)$. 

$$
\hat{v} = \arg \min_v \left\{ \tilde{g}(v) + \frac{a}{2} ||\hat{x} - v + u||^2 \right\} \\
= \arg \min_{v \in \mathbb{R}^+ N} \left\{ \frac{a}{2} ||\hat{x} + u - v||^2 \right\} \\
= \text{clip}\{\hat{x} + u, \mathbb{R}^+ N\} ,
$$
Split-Variable ADMM for Positivity Constraint

Initialize $u \leftarrow 0$ and $\hat{v} \leftarrow 0$

For $K$ iterations {
\begin{align*}
\hat{x} & \leftarrow \arg\min_x \left\{ f(x) + \frac{a}{2} \| x - (\hat{v} - u) \|^2 \right\} \\
\hat{v} & \leftarrow \text{clip}\{ \hat{x} + u, \mathbb{R}^+\} \\
u & \leftarrow u + (\hat{x} - \hat{v})
\end{align*}
}

So this split-variable ADMM algorithm provides a simple and effective method for enforcing positivity in any convex optimization problem. As a practical matter, it can also be used in non-convex optimization problems, but its theoretical convergence properties to a global minimum are no longer guaranteed in this case.

9.2 The Lagrangian and Dual Function

While the examples of the previous section provided intuitive motivation, the discussion was primarily based on “hand waving” arguments. The objective of this section is to provide a rigorous development of the theoretical machinery for constrained optimization, so that the discussion can be put on a much more solid footing. In particular, we will introduce the concept of constrained optimization with Lagrange multipliers along with the concept of a dual function. The dual function will serve a crucial role in selecting the correct value of the Lagrange multiplier that causes the desired constraint to be met.

We will develop this basic theory of constrained optimization by considering problems with the general form

$$x^* = \arg\min_{x \in \Omega} f(x),$$

(9.6)

where $b \in \mathbb{R}^K$, $C \in \mathbb{R}^{K \times N}$, $\Omega \subset \mathbb{R}^N$ is assumed to be a closed convex set with nonempty interior, and $f(x)$ takes values in $\mathbb{R}$. If we would like to enforce positivity in the result, we can set $\Omega = \mathbb{R}^{+N}$, which is a particular example of a closed convex set with a non-empty interior.
In this case, equation (9.6) represents a convex optimization problem. Of course, this is not the most general form of the constrained optimization problem. For example, the constraint could be non-linear with the form $C(x) = b$, or the function $f(x)$ might be non-convex. Nonetheless, the problem of equation (9.6) covers a wide range of important practical problems including linear programming.

The form of equation (9.6) implicitly assumes that the global minimum exists and is unique. However, this may not be the case, even when $f(x)$ is a convex function. In order to handle this more general case, we may reformulate the constrained optimization problem as

$$p^* = \inf_{x \in \Omega} f(x),$$

(9.7)

where $p^* \in [-\infty, \infty)$ is the infimum of the function subject to the constraints. Then if the function takes on its minimum, we have that $p^* = f(x^*)$.

The classic approach to constrained optimization is the method of Lagrange multipliers. The Lagrange method converts a constrained problem to an unconstrained one by adding a linear term that drives the solution to the constraint. To do this, we first define the Lagrangian for the constrained optimization problem as

$$L(x; \lambda) = f(x) + \lambda^t(Cx - b),$$

(9.8)

where $\lambda \in \mathbb{R}^K$ is the so-called Lagrange multiplier; we then compute the solution to the unconstrained optimization problem as

$$x(\lambda) = \arg \min_{x \in \Omega} L(x; \lambda),$$

(9.9)

where $x(\lambda)$ denotes a solution for a specific value of $\lambda$. Of course, the difficulty with this approach is the selection of the correct value for $\lambda$. On the simplest level, the value of $\lambda$ must be chosen so that the constraint is met.

$$Cx(\lambda) = b$$

(9.10)

However, this is typically not a practical approach since it is often difficult to analytically or numerically solve this equation.

---

1 The operations $a = \inf_{x \in \Omega} f(x)$ and $b = \sup_{x \in \Omega} f(x)$ refer to the greatest lower bound and least upper bound, respectively. The infimum and supremum always exist for any real-valued function $f(x)$ and any set $\Omega$, but the results, $a$ and $b$, take values in the extended real line $[-\infty, \infty]$. 
An alternative and very powerful view of the Lagrangian problem comes by considering its dual. We define the dual function as

\[ g(\lambda) = \inf_{x \in \Omega} L(x; \lambda) , \tag{9.11} \]

where \( g : \mathbb{R}^K \to [\mathbb{R}, \infty) \). Notice that for some values of \( \lambda \), it may be that \( g(\lambda) = -\infty \). However, this won’t cause any problems. In this context, we refer to the original function, \( f(x) \), as the primal function and \( g(\lambda) \) as its dual.

As it turns out, the dual function has many interesting and useful properties. First notice, that for each distinct value of \( x \), the function

\[ L(x; \lambda) = f(x) + \lambda^t(Cx - b) , \]

is an affine function of \( \lambda \), which is of course a concave function of \( \lambda \). Therefore by applying Property [A.4] we know that \( g(\lambda) \) must be concave because it is formed from the minimum of a set of concave functions. Notice that this concavity property always holds even when the primal function, \( f(x) \), is not convex.

\textbf{Property 9.1. Concavity of the dual function} - The dual function \( g(\lambda) \) from equation (9.11) is concave on \( \lambda \in \mathbb{R}^K \).

The dual function is very important because it is a lower bound that can be maximized to find the best value for the Lagrange multiplier \( \lambda \). In order to make this concept more formal, define \( d^* \) to be the supremum of the dual function,

\[ d^* \triangleq \sup_{\lambda \in \mathbb{R}^K} g(\lambda) . \tag{9.12} \]

Furthermore, let \( \lambda^* \) denote a value of the Lagrange multiplier that achieves that maximum. So when \( \lambda^* \) exists, then \( d^* = g(\lambda^*) \).

Using these definitions, we next define the duality gap as \( p^* - d^* \), i.e., the difference between the minimum of the primal cost function and the maximum of its dual. We first prove that this duality gap must be positive.

\textbf{Property 9.2. Non-negative duality gap} - The duality gap is non-negative so that

\[ p^* \geq d^* . \]
Proof. The duality gap is a direct result of the very general \textbf{max-min inequality}. (See problem \[3\]) Applying this inequality to the Lagrangian function, we get that

\[ \inf_{x \in \Omega} \sup_{\lambda \in \mathbb{R}^K} L(x; \lambda) \geq \sup_{\lambda \in \mathbb{R}^K} \inf_{x \in \Omega} L(x; \lambda). \] \hspace{1cm} (9.13)

Now in order to better understand the left-hand side of the equation, first notice that

\[ \sup_{\lambda \in \mathbb{R}^K} L(x; \lambda) = \begin{cases} f(x) & \text{if } Cx = b \\ \infty & \text{if } Cx \neq b \end{cases}. \]

So therefore, it must be that

\[ \inf_{x \in \Omega} \sup_{\lambda \in \mathbb{R}^K} L(x; \lambda) = p^*. \]

Similarly, on the right-hand side of the equation, we have that

\[ \inf_{x \in \Omega} L(x; \lambda) = g(\lambda). \]

So substituting these two results into equation (9.13), we have that

\[ p^* \geq \sup_{\lambda \in \mathbb{R}^K} g(\lambda) = d^*, \]

and the proof is complete. \hfill \Box

9.2.1 Strong Duality

In typical cases the dual function is not just a lower bound to the solution of the primal function. In fact, the solutions to the primal and dual functions occur at that same place and take on the same value. This makes the two problems essentially equivalent. We can make this idea more precise by first defining the idea of \textbf{strong duality}. Strong duality holds when \( p^* = d^* \), so that the minimum of the primal function is equal to the maximum of its dual. In the remainder of this section we will see that when strong duality holds, the dual and primal problems can be used together to solve the constrained optimization. This is good news because we will also see that strong duality holds for most practical problems.

The next property states the key result that when strong duality holds, then the unconstrained optimization problem of equation (9.9) can be used
to solve the constrained optimization of equation (9.6). However, to do this, we must set the Lagrange multiplier to $\lambda^*$, the solution to the dual problem of equation (9.12).

**Property 9.3. Optimal Lagrange multiplier** - Consider a constrained optimization problem with the form of equation (9.7) such that:

- Strong duality holds with $p^* = d^*$;
- $x^* \in \Omega$ achieves the minimum to the constrained optimization problem;
- $\lambda^* \in \mathbb{R}^K$ achieves the maximum of the dual function.

Then $x^*$ is also a solution to the unconstrained optimization problem

$$L(x^*; \lambda^*) = \inf_{x \in \Omega} L(x; \lambda^*).$$

Furthermore, if $f(x)$ is strictly convex, then $x^*$ is unique.

**Proof.** Now if $x^*$ is the solution to the constrained optimization problem of equation (9.6), then we know that $Cx^* - b = 0$. Therefore, when strong duality holds, we must have that

$$L(x^*; \lambda^*) = f(x^*) + \lambda^*(Cx^* - b) = f(x^*) = p^* = d^* = g(\lambda^*) = \inf_{x \in \Omega} L(x; \lambda^*).$$

So therefore we know that, when strong duality holds, then a solution to the constrained minimization problem, $x^*$, must also be a solution to the unconstrained optimization problem when $\lambda^*$ is used as the Lagrange multiplier. Moreover, if $f(x)$ is strictly convex, then this solution to the unconstrained optimization problem must be unique.

\[\square\]

**9.2.2 Technical Conditions for Strong Duality**

Property 9.3 tells us that if strong duality holds, then we can solve the constrained optimization problem of equation (9.6) by first computing $\lambda^*$, as
the solution to the dual problem, and then using $\lambda^*$ as the correct Lagrange multiplier in unconstrained optimization of equation (9.9).

As it turns out, strong duality holds in most practical situations. However, finding conditions that ensure this can be a challenge. We can develop more intuition for this problem by first considering the simplified case when both the primal function and its solution are continuously differentiable. In this highly regular case, the gradient of the dual function is given by

$$\nabla g(\lambda^*) = Cx(\lambda^*) - b,$$

where the function $x(\lambda)$ is defined in equation (9.9). (See problem 4) Using this result and the fact that $g(\lambda)$ is concave, we know that $\lambda^*$ is the global maximum of $g(\lambda)$ if and only if $\nabla g(\lambda^*) = Cx(\lambda^*) - b = 0$. From this we can then show that $d^* \geq p^*$ through the following set of inequalities.

$$d^* = g(\lambda^*) = \min_{x \in \Omega} \{ f(x) + [\lambda^*]^t(Cx - b) \}$$

$$= f(x(\lambda^*)) + [\lambda^*]^t(Cx(\lambda^*) - b)$$

$$\geq f(x^*) = p^*$$

Putting this together with Property 9.2, we must have that $d^* = p^*$ and strong duality holds. So we see that strong duality holds when the function $f(x)$ is sufficiently regular on $\mathbb{R}^N$.

Fortunately, there exists a relatively general set of technical conditions known as Slater’s conditions that can be used to ensure strong duality for just about any reasonable convex optimization problem [62, 15]. The following definition states a simplified form of Slater’s conditions for our particular problem.

**Definition 11. Slater’s Conditions**

The optimization problem with the form of equation (9.7) is said to meet Slater’s conditions if there exists an $x'$ in the interior of $\Omega$ such that $Cx' = b$.

Stated differently, Slater’s conditions require that there is at least one strictly feasible solution to the optimization problem. We say that the solution is strictly feasible because it falls in the interior of $\Omega$ and it meets
the constraint of $Cx' = b$. For most practical problems, it is easy to find a feasible solution that falls within the interior of the domain. So typically, this is a technical constraint that can be easily verified.

The following theorem proved by Slater provides a powerful tool for ensuring that strong duality holds in practical situations.

**Property 9.4. Slater’s Theorem** - Consider a constrained optimization problem with the form of equation (9.7) such that Slater’s conditions hold and $f(x)$ is a convex function on a closed convex set $\Omega$. Then

- Strong duality holds with $p^* = d^*$;
- There exists $\lambda^* \in \mathbb{R}^K$ such that $d^* = g(\lambda^*)$.

\[
\lambda^* = \arg \max_{\lambda \in \mathbb{R}^K} g(\lambda) .
\] (9.14)

Then we use the resulting value of $\lambda^*$ to solve the unconstrained optimization problem

\[
x^* = \arg \min_{x \in \Omega} \left\{ f(x) + [\lambda^*]^t (Cx - b) \right\} ,
\] (9.15)

where $x^*$ is then the solution to the constrained optimization problem of equation (9.6). However, the difficulty with this approach is that the dual function, $g(\lambda)$, typically does not have closed form. So it is still not clear how to determine $\lambda^*$. The following section provides a practical solution to this problem.
9.3 The Augmented Lagrangian

Lagrange multipliers provide a beautiful theory for solving constrained optimization problems. However, at this point, it is unclear how to practically compute the value of $\lambda^*$ that will properly enforce the constraint. In some analytical problems, it is possible to explicitly solve the constraint of equation (9.10) for the desired value of $\lambda^*$. However, for large numerical problems, this is not possible. Another approach is to determine $\lambda^*$ by maximizing the dual function of $g(\lambda)$. However, the function $g(\lambda)$ usually does not have an explicit form that can be easily maximized. So at this point, the framework of Lagrange multipliers and primal-dual functions seems beautiful but perhaps somewhat useless.

Fortunately, the augmented Lagrangian method [37, 54] solves this problem by providing a numerically stable approach for computing $\lambda^*$ by simultaneously solving both the primal and dual problems. More specifically, we will see that the augmented Lagrangian approach results in an explicated algorithm that is easily implemented and that is theoretically guaranteed to converge to the optimal Lagrange multiplier $\lambda^*$.

The augmented Lagrangian approach works by augmenting the basic optimization problem with an additional quadratic term that helps drive the solution towards the constraint. However, an unexpected result of adding this additional term is that we can then derive a recursion for $\lambda$ that converges to the desired value of $\lambda^*$.

Now, in order to derive the methods and theory of augmented Lagrangian optimization, we will first define the following canonical convex optimization problem as the basis for our results,

$$p^* = \inf_{x \in \Omega} f(x),$$

(9.16)

with the following properties:

- $f : \Omega \to \mathbb{R}$ is a convex function on a closed convex set $\Omega \subset \mathbb{R}^N$;
- Slater’s conditions hold;
- there exists a solution $x^* \in \Omega$ such that $p^* = f(x^*)$ and $C x^* = b$. 
The associated dual function for this problem is then given by
\[ g(\lambda) = \inf_{x \in \Omega} \left\{ f(x) + \lambda^t(Cx - b) \right\} . \] (9.17)

By Property [9.4], we know that the dual function takes on its maximum.
\[ \lambda^* = \arg \max_{\lambda \in \mathbb{R}^K} g(\lambda) . \] (9.18)

Using this framework, we first define a cost function for the augmented Lagrangian as
\[ L(x; a, \lambda) = f(x) + \lambda^t(Cx - b) + \frac{a}{2}||Cx - b||^2 , \] (9.19)
where the additional term \( \frac{a}{2}||Cx - b||^2 \) is added in order to drive the solution towards the constraint of \( Cx = b \). Notice that as \( a \) becomes larger, optimization will naturally tend to reduce the value of \( ||Cx - b||^2 \), and therefore enforce the constraint of \( Cx = b \). This would appear to be an effective practical technique to enforce the desired constraint. However, if \( a \) is chosen to be too large, then the optimization problem can become very stiff and convergence will be very slow.

So our goal will be to find a tractable recursion for \( \lambda \) that will produce the correct value of the Lagrange multiplier. To achieve this goal, we will first prove the following very valuable lemma.

**Lemma 1.** Consider the convex optimization problem of (9.16) with dual function of (9.17) and augmented Lagrangian of (9.19). Then for all \( \lambda \in \mathbb{R}^K \)
\[ \inf_{x \in \Omega} L(x; a, \lambda) = \sup_{\lambda' \in \mathbb{R}^K} \left\{ g(\lambda') - \frac{1}{2a}||\lambda' - \lambda||^2 \right\} , \] (9.20)
where the infimum and supremum are achieved. Furthermore, the values \( x^{**} \in \Omega \) and \( \lambda^{**} \in \mathbb{R}^K \) that achieve the infimum and supremum satisfy the recursion
\[ \lambda^{**} = \lambda + a(Cx^{**} - b) . \] (9.21)

**Proof.** The proof we present is adapted from [5] page 244.

We start by proving that \( x^{**} \) and \( \lambda^{**} \) must exist. First, notice that \( g(\lambda) \) is concave and due to Slater’s conditions, it achieves its maximum. Therefore,
9.3 The Augmented Lagrangian 179

\[ g(\lambda') - \frac{1}{2a} ||\lambda' - \lambda||^2 \] must be strictly concave and must also uniquely achieve its maximum. Next, since \( f(x) \) is convex and \( x^* \) achieves the constrained minimum, the set \( \{x \in \Omega : L(x; a, \lambda) \leq T\} \) must be compact for some \( T \). Therefore, \( L(x; a, \lambda) \) must take on a global minimum.

Next we present a sequence of equalities that prove the equality of equation (9.20).

\[
\inf_{x \in \Omega} L(x; a, \lambda) = \inf_{x \in \Omega} \begin{cases} f(x) + \lambda^t (Cx - b) + \frac{a}{2} ||Cx - b||^2 \end{cases} \\
= \inf_{x \in \Omega, z \in \mathbb{R}^K} \begin{cases} f(x) + \lambda^t z + \frac{a}{2} ||z||^2 \end{cases} \quad \text{subject to } z = Cx - b \\
= \sup_{\lambda'} \inf_{z \in \mathbb{R}^K} \begin{cases} f(x) + [\lambda']^t (Cx - b - z) + \lambda^t z + \frac{a}{2} ||z||^2 \end{cases} \\
= \sup_{\lambda'} \left\{ \inf_{x \in \Omega} \begin{cases} f(x) + [\lambda']^t (Cx - b) \end{cases} + \inf_{z \in \mathbb{R}^K} \begin{cases} (\lambda - \lambda')^t z + \frac{a}{2} ||z||^2 \end{cases} \right\} \\
= \sup_{\lambda'} \left\{ g(\lambda') + \inf_{z \in \mathbb{R}^K} \begin{cases} (\lambda - \lambda')^t z + \frac{a}{2} ||z||^2 \end{cases} \right\}
\]

The first equality follows from the definition of the function \( L(x; a, \lambda) \). The second equality holds by substituting \( z \) for the expression \( Cx - b \) and applying the constraint that \( z = Cx - b \). The third equality holds by introducing a new Lagrange multiplier, \( \lambda' \), to enforce the constraint \( z = Cx - b \) and then applying the strong duality property. The fourth equality holds by collecting terms in \( x \) and \( z \). The fifth equality holds by the definition of \( g(\lambda) \).

Next, we can simplify the expression in the last term by calculating the explicit minimum. This yields the result that

\[
g(\lambda') + \inf_{z \in \mathbb{R}^K} \begin{cases} (\lambda - \lambda')^t z + \frac{a}{2} ||z||^2 \end{cases} = g(\lambda') - \frac{1}{2a} ||\lambda - \lambda'||^2
\]

where it is easily shown that the minimization over \( z \) results when

\[
z = \frac{\lambda' - \lambda}{a}.
\] (9.22)
Using this result, we have that
\[
\inf_{x \in \Omega} L(x; a, \lambda) = \sup_{\lambda'} \left\{ g(\lambda') + \inf_{z \in \mathbb{R}^K} \left\{ (\lambda - \lambda')^t z + \frac{a}{2} ||z||^2 \right\} \right\}
\]
\[
= \sup_{\lambda'} \left\{ g(\lambda') - \frac{1}{2a} ||\lambda - \lambda'||^2 \right\}
\]
\[
= g(\lambda^{**}) - \frac{1}{2a} ||\lambda - \lambda^{**}||^2
\]
where \(\lambda^{**}\) is defined as the value that achieves the maximum. So this proves the first result of the lemma.

Notice that when \(\lambda' = \lambda^{**}\), then equation (9.22) becomes
\[
z^{**} = \frac{\lambda^{**} - \lambda}{a}, \quad (9.23)
\]
where \(z^{**}\) is the associated value of \(z\) when \(\lambda' = \lambda^{**}\). Now, equation (9.23) is the result of solving the constrained optimization with \(z = Cx - b\). So in particular, it must be that \(z^{**} = Cx^{**} - b\) where \(x^{**}\) is the value of \(x\) that achieved the minimum of \(L(x; a, \lambda)\). Substituting this result into (9.22) then yields that
\[
Cx^{**} - b = \frac{\lambda^{**} - \lambda}{a},
\]
and rearranging terms, then yields the final result that
\[
\lambda^{**} = \lambda + a(Cx^{**} - b),
\]
which completes the proof. 

With this lemma in hand, we may now introduce the \textbf{augmented Lagrangian algorithm} as the following conceptually simple recursion in \(k\).
\[
x^{(k+1)} = \arg \min_{x \in \Omega} L(x; a, \lambda^{(k)}) \quad (9.24)
\]
\[
\lambda^{(k+1)} = \lambda^{(k)} + a(Cx^{(k+1)} - b) \quad (9.25)
\]
The optimization step of (9.24) computes the unconstrained minimum of the augmented Lagrangian cost function with respect to \(x\), and the update step of (9.25) computes a very simple update of the Lagrange multiplier based on the error in the constraint.
We can now show that these iterations converge by using Lemma 1. From the lemma, we know that each update of this recursion is equivalent to the following recursion in $\lambda^{(k)}$.

$$
\lambda^{(k+1)} = \arg \max_{\lambda \in \mathbb{R}^K} \left\{ g(\lambda) - \frac{1}{2a} \| \lambda - \lambda^{(k)} \|^2 \right\}
$$

(9.26)

In fact, it is easily shown that this iteration must converge to

$$
\lim_{k \to \infty} \lambda^{(k)} = \lambda^* = \arg \max_{\lambda \in \mathbb{R}^K} g(\lambda).
$$

Intuitively, each iteration attempts to maximize $g(\lambda)$, but is restricted by the term $\frac{1}{2a} \| \lambda - \lambda^{(k)} \|^2$ which “pulls” the solution towards the previous value of $\lambda^{(k)}$. However, after each new iteration, this restriction becomes weaker, and the solution asymptotically converges to the global maximum of $g(\lambda)$. Homework problem 6 presents a proof of convergence.

The following property states this convergence result formally.

**Property 9.5. Convergence of Augmented Lagrangian** - Consider the convex optimization problem of (9.16). Then the augmented Lagrangian algorithm of equations (9.24) and (9.25) converge in the sense that

$$
\lim_{k \to \infty} \lambda^{(k)} = \lambda^*,
$$

(9.27)

where $\lambda^* = \arg \max_{\lambda \in \mathbb{R}^K} g(\lambda)$, and

$$
x^* = \arg \min_{x \in \Omega} L(x; a, \lambda^*),
$$

(9.28)

where $x^*$ is a solution to the constrained optimization of (9.16).

**Proof.** We have already proved the convergence of equation (9.27) to the maximum of the dual function $g(\lambda)$. By a combination of Properties 9.4 and 9.3, we know that strong duality must hold and that any solution to the constrained optimization problem must be a solution to

$$
x^* = \arg \min_{x \in \Omega} L(x; 0, \lambda^*),
$$

where $a = 0$. Since $x^*$ is assumed to meet the constraint, we know that $Cx^* - b = 0$, and therefore that

$$
x^* = \arg \min_{x \in \Omega} L(x; a, \lambda^*),
$$
Augmented Lagrangian Algorithm:

Initialize $u \leftarrow 0$

For $K$ iterations {

$\hat{x} \leftarrow \arg \min_{x \in \Omega} \left\{ f(x) + \frac{a}{2} ||Cx - b + u||^2 \right\}$

$u \leftarrow u + C\hat{x} - b$.

}

Figure 9.1: Pseudocode for the implementation of the augmented Lagrangian algorithm for the minimization of $f(x)$ subject to the constraint $Cx = b$. This iteration can be shown to converge to the optimal value of the Lagrange multiplier $u^*$ that achieves the desired constraint.

which completes the proof.

Finally, we can reorganize the terms of the augmented Lagrangian to put it in an equivalent but slightly more compact form.

$$x^{(k+1)} = \arg \min_{x \in \Omega} \left\{ f(x) + \frac{a}{2} ||Cx - b + u^{(k)}||^2 \right\} \quad (9.29)$$

$$u^{(k+1)} = u^{(k)} + Cx^{(k+1)} - b. \quad (9.30)$$

It is easy to show that this new form is equivalent to the form of equation (9.19) by expanding out the expression for $\frac{a}{2} ||Cx - b + u^{(k)}||^2$. (See problem 7) In the new form, the vector $u$ serves the role of the Lagrange multiplier but it is scaled so that $u = \lambda/a$. Figure 9.1 shows the resulting algorithm in pseudocode form.

It is interesting to note that the convergence of the augmented Lagrangian algorithm does not depend on the choice of $a$. In fact, a very large value of $a$ will result in much faster convergence in terms of number of iterations. This can be seen in two ways. First, if $a$ is large, then the term $\frac{a}{2} ||Cx - b + u^{(k)}||^2$ will drive the solution towards the constraint more rapidly. Alternatively, when $\frac{1}{2a}$ is small, then the dual-space iteration of equation (9.26) will be less restricted by the term $\frac{1}{2a} ||\lambda - \lambda^{(k)}||^2$. However, the price one pays for this fast convergence is that each iteration can be more difficult. This is because when $a$ is very large, then the optimization of equation (9.29) becomes more ill-conditioned and therefore, more difficult. So the trick then becomes to balance these two objectives of fast convergence and well-conditioned minimization.
9.4 Shrinkage and Proximal Maps

At this point, we take a minor detour to better understand a rather simple operation that reappears in a number of different contexts. Let \( u(x) \) be a convex function on the closed convex nonempty set \( \Omega \subset \mathbb{R}^N \). Then we define the proximal map, \( H(y) \) of \( u(x) \) as

\[
H(y) = \arg \min_{x \in \Omega} \left\{ \frac{1}{2} \| y - x \|^2 + u(x) \right\}.
\]  

(9.31)

Figure 9.2: The graphic illustration of a proximal map. The input, \( y \), can be thought of as a stake in the ground holding one end of a spring. The other end of the spring with unit spring constant holds a cart on wheels that would otherwise roll down the hill of the function being minimized. The cart moves toward the bottom of the function \( f(x) \), but it is restricted by the spring, so instead it reaches an equilibrium point where the force of the hill balances the force of the spring.

Moreover, we note that the proximal mapping of equation (9.31) has another simple interpretation. Consider the problem in which \( Y = X + W \)

\footnote{2 More generally, \( u \) may be an extended convex function in which case \( u(x) \) may be any proper closed convex function \( f : \mathbb{R}^N \to \mathbb{R} \cup \{\infty\} \).}

\footnote{3 This physical analogy is not exact because in this case the spring extends diagonally rather than horizontally as the proximal map definition requires. But the emotional intuition you get by looking at the figure is correct.}
where $W$ is white Gaussian noise with unit variance and our objective is to recover $X$ from $Y$. This simplest of inverse problems, known as denoising, has been the focus of enormous attention over the years. Perhaps this is due to the fact that denoising is not only of practical importance but also that it presents a pure theoretical problem in image modeling. Expanding on this idea, the MAP cost function for this denoising problem is given by

$$f(x; y) = \frac{1}{2}||y - x||^2 + u(x)$$

where $u(x) = -\log p(x)$ is the convex negative log prior distribution. From this, we see that the MAP estimate, $\hat{x} = H(y)$, is given by the proximal mapping of (9.31). It is also interesting to note that by substituting in $u(x) = -g(x)$, then the proximal mapping has the same structure as the update equation for the augmented Lagrangian in equation (9.26).

Next, we make some useful observations about the proximal mapping. First, notice that $f(x; y)$ is a strictly convex function on $\Omega$. Therefore, the global minimum must be unique, and $H(y)$ must always be well defined for $y \in \mathbb{R}^N$. Also, in homework problem 6, we show that iterative application of a proximal mapping converges to the global minimum of the function $u(x)$. So the recursion

$$y^{(k+1)}(k+1) = \arg \min_{x \in \Omega} \left\{ \frac{1}{2}||y^{(k)} - x||^2 + u(x) \right\} , \quad (9.32)$$

converges to the minimum of $u(x)$, or more formally

$$\lim_{k \to \infty} y^{(k)} = \arg \min_x u(x) .$$

In retrospect, this is the same reason that the augmented Lagrangian iteration of equation (9.26) is guaranteed to converge to the solution of the dual optimization problem.

We can gain insight into the value of the proximal mapping by considering some illustrative special cases. One very interesting case occurs when we use the $L_1$ prior first discussed in Section 6.3 as the regularizing prior. In this case, $u(x) = \lambda||x||_1$ where $\lambda \geq 0$ is a scalar parameter, and the proximal mapping is given by

$$H(y) = \arg \min_{x \in \mathbb{R}^N} \left\{ \frac{1}{2}||y - x||^2 + \lambda||x||_1 \right\} . \quad (9.33)$$
Once again, since $||x||_1$ is a convex function, we know that the function being minimized is a strictly convex function of $x$ and therefore has a unique global minimum. We can calculate the exact minimum by first noticing that the function being minimized is separable, so that it can be written as the sum of independent terms.

$$f(x; y) = \sum_{i=1}^{N} \left\{ \frac{1}{2} (y_i - x_i)^2 + \lambda |x_i| \right\}$$

For each component, $y_i$, the solution can therefore be computed by independently minimizing with respect to each component, $x_i$. Minimizing with respect to $x_i$ results in the following solution.

$$\arg \min_{x_i \in \mathbb{R}} \left\{ \frac{1}{2} (y_i - x_i)^2 + \lambda |x_i| \right\} = \begin{cases} y_i - \lambda & \text{for } y_i > \lambda \\ 0 & \text{for } |y_i| \leq \lambda \\ y_i + \lambda & \text{for } y_i < -\lambda \end{cases} \quad (9.34)$$

This result can be easily verified by considering the three cases. (See problem 8)

The expression of equation (9.34) occurs so commonly that it has been named the shrinkage function. With a little bit of manipulation, we may express the shrinkage function in a more compact form given by

$$S_\lambda(y_i) \triangleq \text{sign}(y_i) \max \{ |y_i| - \lambda, 0 \} , \quad (9.35)$$

where $S_\lambda(y_i)$ is defined as the shrinkage function applied to the scalar $y_i$. Using this more compact notation, we can also express the solution to the proximal mapping of equation (9.33) as

$$\arg \min_{x \in \mathbb{R}^N} \left\{ \frac{1}{2} ||y - x||^2 + \lambda ||x||_1 \right\} = S_\lambda(y) , \quad (9.36)$$

where $S_\lambda(y)$ is now interpreted as applying the shrinkage function to each component of the vector $y$.

Figure 9.3 graphically illustrates the form of the shrinkage function. Notice that it is a continuous function, and that values of $y_i$ with magnitude $\leq \lambda$ are set to zero. Other values of $y_i$ are “shrunk” towards zero, with positive values being moved down, and negative values being moved up. Intuitively for the denoising case, if the value of $|y_i|$ is below $\lambda$, then it seems likely that
the component contains mostly noise; so the value is set to zero. However, if the value of $|y_i|$ is above $\lambda$, then it seems likely that the component contains mostly signal and it is retained, but with attenuation.

In fact, this framework extends to the more general case when our inverse problem is based on the model $Y = AX + W$ where $A$ is an $M \times N$ matrix and $W \sim N(0, \Lambda^{-1})$. If we assume that the components of $X$ are i.i.d. with a Laplacian distribution, then the prior density is given by

$$p_\lambda(x) = \left(\frac{\lambda}{2}\right)^N \exp\left\{-\lambda||x||_1\right\},$$

and the corresponding MAP optimization problem is then given by

$$\hat{x} = \arg\min_{x \in \mathbb{R}^N} \left\{\frac{1}{2}||y - Ax||_A^2 + \lambda||x||_1\right\}. \quad (9.37)$$

Notice that this MAP estimate has a very similar form to the proximal map of equation (9.36). So perhaps it is not surprising that the ICD update equation for solving this problem will incorporate the shrinkage function. In order to see this, we may use equation (7.13), to rewrite the MAP cost function as a function of the single pixel in the form

$$f_s(x_s) = \theta_1(x_s - x_{old}) + \frac{1}{2}\theta_2(x_s - x_{old})^2 + \lambda|x_s| + \text{const}$$
where \( x_{\text{old}} \) is the previous value of \( x_s \), and \( \theta_1 \) and \( \theta_2 \) are given in equations (7.14) and (7.15) as

\[
\begin{align*}
\theta_1 &= -(y - Ax)^t \Lambda A_{*,s} \\
\theta_2 &= A_{*,s}^t \Lambda A_{*,s}.
\end{align*}
\]

Using this simplified expression, it is possible to write the ICD update as

\[
x_s \leftarrow \arg \min_{x_s} \left\{ \frac{1}{2} \left( x_s - \left[ x_{\text{old}} - \frac{\theta_1}{\theta_2} \right] \right)^2 + \frac{\lambda}{\theta_2} |x_s| \right\}.
\]

Notice that this update now has the form of the shrinkage operator given in equation (9.34); so the ICD update can be written as

\[
x_s \leftarrow S_{\lambda/\theta_2} \left( x_{\text{old}} - \frac{\theta_1}{\theta_2} \right).
\]

Figure 9.4 shows the pseudocode algorithm for computing the ICD updates of this \( L_1 \) regularized optimization problem of equation (9.37). Interestingly, in this case the ICD algorithm is provably convergent to the global minimum even though the MAP cost function is not continuously differentiable and therefore does not meet the conditions of Theorem 7.2.1. Intuitively, the ICD algorithm converges in this case because the updated directions of the ICD algorithm are perpendicular to the discontinuous manifolds of the \( L_1 \) norm that occur when \( x_i = 0 \) for any \( i \).

However, in the more general problem with the form

\[
\hat{x} = \arg \min_{x \in \mathbb{R}^N} \left\{ \frac{1}{2} ||y - Ax||_A^2 + ||Bx||_1 \right\},
\]

where \( B \) is an \( N \times N \) matrix, then the ICD updates are not in general convergent to the global minimum. This means that the ICD updates cannot be used to guarantee a solution to important problems such as the total-variation regularization problem of Section 6.3.

Fortunately, in the Section 9.5.1, we will introduce a very clever method for constrained optimization that can be used to solve problems with the form of equation (9.38).
ICD Algorithm using $L_1$ Prior:

Initialize $e \leftarrow y - Ax$

For $K$ iterations {
  For each pixel $s \in S$ {
    $x_{old} \leftarrow x_s$
    $\theta_1 \leftarrow -e^t \Lambda A_{*,s}$
    $\theta_2 \leftarrow A^t_{*,s} \Lambda A_{*,s}$
    $x_s \leftarrow S_{\lambda/\theta_2} \left( x_{old} - \frac{\theta_1}{\theta_2} \right)$
    $e \leftarrow e - A_{*,s}(x_s - x_{old})$.
  }
}

Figure 9.4: Pseudocode for fast implementation of the ICD algorithm with an $L_1$ prior as shown in equation (9.37). The algorithm is guaranteed to converge to the global minimum of the convex cost function.

9.5 Variable Splitting and the ADMM Algorithm

In this section, we will show how constrained optimization can be used to solve a number of practical problems in MAP optimization through the use of two synergistic methods: Variable splitting and the ADMM algorithm [23, 25]. In fact, the two methods were informally introduced in Section 9.1 in the context of enforcing positivity constraints. However, in this section, we will reexamine these two methods in a bit more general context.

So for example, consider any optimization problem with the general form

$$\hat{x} = \arg\min_{x \in \Omega} \{ f(x) + g(Bx) \},$$

(9.39)

where $B$ is a $K \times N$ matrix, and $f(x)$ and $g(v)$ are convex functions.

Now to solve this problem, we will split the variable $x$ into two variables, $x$ and $v$, with the constraint that $Bx = v$. So then the new problem is given by

$$\langle \hat{x}, \hat{v} \rangle = \arg\min_{\langle x, v \rangle \text{ s.t. } Bx = v} \{ f(x) + g(v) \}.$$  

(9.40)

Given the constraint it must be that $\hat{v} = B\hat{x}$, so therefore that the solution to equations (9.39) and (9.40) must be the same.
Using the augmented Lagrangian methods of Section 9.3, we can convert this constrained optimization problem to be an unconstrained optimization problem with the form

\[
(\hat{x}, \hat{v}) = \arg \min_{(x,v)} \left\{ f(x) + g(v) + \frac{a}{2} \|Bx - v + u\|^2 \right\},
\]  
(9.41)

where \( u \) must be chosen to achieve the desired constraint that \( B\hat{x} = \hat{v} \). Now in order to determine the correct value of the Lagrange multiplier \( u \), we will use the augmented Lagrangian algorithm of Figure 9.1. This results in the following iteration.

Repeat\{
(\hat{x}, \hat{v}) \leftarrow \arg \min_{(x,v)} \left\{ f(x) + g(v) + \frac{a}{2} \|Bx - v + u\|^2 \right\}  
(9.42)
\]
\[ u \leftarrow u + B\hat{x} - \hat{v} \]  
(9.43)
\}

Now let us focus on solving the optimization of equation (9.42). In order to solve this optimization, we can use the method of alternating minimization specified by the following iteration.

Repeat\{
\[
\hat{x} \leftarrow \arg \min_{x} \left\{ f(x) + \frac{a}{2} \|Bx - \hat{v} + u\|^2 \right\} 
(9.44)
\]
\[
\hat{v} \leftarrow \arg \min_{v} \left\{ g(v) + \frac{a}{2} \|B\hat{x} - v + u\|^2 \right\} 
(9.45)
\]
\}

Notice that we drop the term \( g(v) \) in equation (9.44) since it is not a function of \( x \), and we drop the term \( f(x) \) in equation (9.45) since it is not a function of \( v \). Since this alternating iteration can be a form of coordinate descent from Theorem 7.2.1, we know that this iteration should converge to the desired solution of equation (9.5).

Going back to the problem of the augmented Lagrangian, we must embed this alternating minimization within each iteration of the augmented Lagrangian algorithm. This would result in a nested iteration, which while possible, can be computationally expensive. Instead, we trust our instincts
ADMM algorithm for minimizing \( f(x) + g(Bx) \):

Initialize \( a > 0 \), \( u \leftarrow 0 \), and \( \hat{v} \leftarrow 0 \)

For \( K \) iterations

\[
\hat{x} \leftarrow \arg \min_{x \in \Omega} \left\{ f(x) + \frac{a}{2} \| Bx - (\hat{v} - u) \|^2 \right\} \tag{9.46}
\]

\[
\hat{v} \leftarrow \arg \min_{v \in \mathbb{R}^K} \left\{ g(v) + \frac{a}{2} \| v - (B\hat{x} + u) \|^2 \right\} \tag{9.47}
\]

\[
u \leftarrow u + B\hat{x} - \hat{v} \tag{9.48}
\]

Figure 9.5: Pseudocode for ADMM optimization of equation (9.39). The first two equations provide alternating optimization of the variables \( x \) and \( v \), while the last equation updates \( u \) according to the augmented Lagrangian method to enforce the constraint that \( Bx = v \).

as good irreverent engineers, and we assume that one iteration of the alternating minimization should be “enough”. Perhaps surprisingly, our unsavory engineering instincts are quite correct in this case. In fact, the algorithm that results from a single iteration of alternating minimization is the ADMM algorithm, and it can be proved that ADMM still has guaranteed convergence [14, 19].

Figure 9.5 shows the general form of the ADMM algorithm for the problem of equation (9.39). The algorithm consists of three updates performed repeatedly until convergence. The first update of equation (9.46) minimizes the augmented Lagrangian cost function with respect to the variable \( x \). The second update of equation (9.47) does the same for \( v \), and the third of equation (9.48) updates the variable \( u \) of the augmented Lagrangian in order to enforce the required constraint that \( v = Bx \). One subtle but important point is that the optimization with respect to \( v \) no longer requires the enforcement of the constraint \( x \in \Omega \). As we will soon see, this can provide an important simplification in some circumstances.

So we see that the ADMM algorithm results from a series of three tricks:

- Split a single variable into two variables in order to separate the two terms in the optimization
- Apply the augmented Lagrangian method to enforce consistency of the
ADMM algorithm for Total-Variation Regularization:

Initialize $a > 0$, $u \leftarrow 0$, and $\hat{v} \leftarrow 0$

For $K$ iterations {
\begin{align}
\hat{x} &\leftarrow \arg \min_{x \in \mathbb{R}^N} \left\{ \frac{1}{2}||y - Ax||^2_\Lambda + \frac{a}{2}||Bx - \hat{v} + u||^2 \right\} \quad (9.49) \\
\hat{v} &\leftarrow S_{1/a}(Bx + u) \quad (9.50) \\
u &\leftarrow u + B\hat{x} - \hat{v} \quad (9.51)
\end{align}
}

Figure 9.6: Pseudocode for ADMM optimization of the total-variation regularization problem from equation (9.52). The first two equations provide alternating optimization of the variables $x$ and $v$, while the last equation updates $u$ according to the augmented Lagrangian method to enforce the constraint that $Bx = v$.

split variables

- Use a single iteration of alternating minimization to minimize with respect to each of the split variables

Based on this view, it is clear that ADMM can be used for wide range of applications.

Intuitively, the ADMM algorithm has many practical advantages. First, it allows different optimization methods to be used with different parts of the cost function. Second, the augmented Lagrangian method allows the constraint to be relaxed during the optimization process, but ultimately enforces it at convergence. Intuitively, this flexibility to relax the constraint during optimization can allow the optimization algorithm to converge more efficiently to its solution.

### 9.5.1 Total Variation Regularization using ADMM

In this section, we will show how variable splitting with ADMM can be applied to efficiently solve the problem of total-variation regularized inversion of equation (9.38).

For this problem, the MAP cost function is given by

$$f(x; y) = \frac{1}{2}||y - Ax||^2_\Lambda + ||Bx||_1,$$ (9.52)
where $A$ is a $M \times N$ forward matrix, $B$ is a general $K \times N$ matrix, and $\|z\|_1 = \sum_i |z_i|$ is the $L_1$ norm of $z$. As we learned in Theorem 7.2.1, the convergence of ICD optimization depends critically on the MAP cost function being continuously differentiable. In fact, most gradient based optimization techniques depend on continuity of the derivative in order to achieve convergence. So the computation of the MAP estimate in this case seems to be problematic.

To do this, we convert the unconstrained optimization over the single vector $x$ into an optimization over two vectors $x \in \mathbb{R}^N$ and $v \in \mathbb{R}^K$ with the constraint that $v = Bx$. So the problem becomes

$$(\hat{x}, \hat{v}) = \arg\min_{(x,v)} \left\{ \frac{1}{2} \| y - Ax \|_\Lambda^2 + \|v\|_1 \right\}.$$ \hspace{1cm} (9.53)

From this, we can formulate the associated augmented Lagrangian to yield the following iteration.

Repeat \{ 
$$(\hat{x}, \hat{v}) \leftarrow \arg\min_{(x,v)} \left\{ \frac{1}{2} \| y - Ax \|_\Lambda^2 + \|v\|_1 + \frac{a}{2} \|Bx - v + u\|_2^2 \right\}$$ \hspace{1cm} (9.54)

$u \leftarrow u + B\hat{x} - \hat{v}$$

\}

Now in order to implement the ADMM algorithm, we must be able to solve the optimization of equation (9.54) using alternating minimization of $x$ and $v$. The function is quadratic in $x$; so minimization with respect to this variable is straightforward. Minimization with respect to $v$ yields the following update formula.

$$v \leftarrow \arg\min_{v \in \mathbb{R}^K} \left\{ \frac{1}{2} \| v - (Bx + u) \|_2^2 + \frac{1}{a} \|v\|_1 \right\}$$

So here we see that optimization with respect to $v$ is exactly a proximal mapping with precisely the same form as given in equation (9.36). Therefore, we can express the solution in terms of the shrinkage function.

$$v \leftarrow S_{1/a}(Bx + u)$$ \hspace{1cm} (9.55)
ADMM algorithm for Minimization with Convex Constraint:

Initialize $a > 0$, $u \leftarrow 0$, and $\hat{v} \leftarrow 0$

For $K$ iterations {

\[
\hat{x} \leftarrow \text{arg min}_{x \in \mathbb{R}^N} \left\{ f(x) + \frac{a}{2} \| x - \hat{v} + u \|^2 \right\} \quad (9.58)
\]

\[
\hat{v} \leftarrow \text{clip}(\hat{x} + u, A) \quad (9.59)
\]

\[
u \leftarrow u + \hat{x} - \hat{v} \quad (9.60)
\]

Figure 9.7: Pseudocode for ADMM minimization of a convex function $f(x)$ with a positivity constraint on $x$. The first two equations provide alternating optimization of the variables $x$ and $v$, while the last equation updates $u$ according to the augmented Lagrangian method to enforce the constraint that $v = x$. Importantly, equation (9.58) only requires unconstrained optimization of $x$.

Putting this together with the quadratic update of $x$, we know that the following repetition of alternating minimizations should converge to the global minimum of equation (9.54).

Repeat{

\[
\hat{x} \leftarrow \text{arg min}_{x \in \mathbb{R}^N} \left\{ \frac{1}{2} \| y - Ax \|_A^2 + \frac{a}{2} \| Bx - \hat{v} + u \|^2 \right\} \quad (9.56)
\]

\[
\hat{v} \leftarrow S_{1/a}(B\hat{x} + u) \quad (9.57)
\]

Figure 9.6 shows the ADMM algorithm for total-variation regularization that results from a single iteration of the alternating minimization. Again, this algorithm is guaranteed to converge based on theory discussed in Section 9.5.3.

### 9.5.2 Enforcing Convex Constraints using ADMM

The ADMM algorithm is particularly useful for enforcing convex constraints such as positivity. In order to illustrate this, consider the general problem of

\[
\hat{x} = \arg \min_{x \in \mathcal{A}} f(x) \quad , (9.61)
\]
where $\mathcal{A} \subset \mathbb{R}^N$ is a non-empty convex set and $f$ is a convex function. Then following the logic of Section 9.1, we can express this problem in the form

$$\hat{x} = \arg \min_x \{ f(x) + \tilde{g}(x) \}, \quad (9.62)$$

where

$$\tilde{g}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{A} \\ \infty & \text{if } x \notin \mathcal{A} \end{cases}$$

is a proper, closed, convex function on $\mathbb{R}^N$.

In order to enforce the constraint, we will first split the variable $x$ into two variables, $x$ and $v$, such that $x = v$. Formally, we then represent the optimization problem as

$$(\hat{x}, \hat{v}) = \arg \min_{(x,v) \in \mathbb{R}^N \times \mathcal{A}} \{ f(x) + \tilde{g}(v) \}.$$

From this, we can derive the associated augmented Lagrangian optimization algorithm.

Repeat

$$(\hat{x}, \hat{v}) \leftarrow \arg \min_{(x,v) \in \mathbb{R}^N \times \mathcal{A}} \left\{ f(x) + \tilde{g}(v) + \frac{a}{2} \| x - v + u \|_2 \right\}, \quad (9.63)$$

$$u \leftarrow u + \hat{x} - \hat{v} \quad (9.64)$$

If we replace the optimization of equation (9.63) with the alternating minimization of $x$ and $v$, we get the resulting ADMM algorithm minimization of $f$ with a convex constraint.

### 9.5.3 Convergence of ADMM Algorithm

We finish this chapter by presenting some formal results for guaranteeing convergence of the ADMM algorithm. Our results are based on [14], but with some additional context that adapts them for imaging applications.

In order to do this, we first define the following canonical constrained optimization,

$$p^* = \inf_{x \in \Omega, \ z \in \Phi} \{ f(x) + h(z) \}, \quad (9.65)$$

where $A \in \mathbb{R}^{K \times N}; B \in \mathbb{R}^{K \times L}; b \in \mathbb{R}^K$ with the following three properties,
The ADMM optimization algorithm for this canonical problem is given in Figure 9.8. Our goal for this section is then to find conditions that guarantee convergence of the ADMM algorithm to the global minimum of the constrained optimization problem. To do this, we first define the concept of a saddle point solution.
Definition 12. Saddle Point Solution

The triple \((x^*, z^*, \lambda^*)\) are said to be a saddle point solution if for all \(x \in \Omega, z \in \Phi\), and for all \(\lambda \in \mathbb{R}^K\), we have that

\[
L(x, z; \lambda^*) \geq L(x^*, z^*; \lambda^*) \geq L(x^*, z^*; \lambda) .
\]

The following property states that any solution to our canonical optimization problem must be a saddle point solution.

Property 9.6. Saddle Point Property - Consider the canonical constrained optimization problem of equation (9.65). Then

- There exists \(\lambda^* \in \mathbb{R}^K\) such that \(d^* = g(\lambda^*)\);
- The triple \((x^*, z^*, \lambda^*)\) is a saddle point solution.

Proof. The canonical problem of equation (9.65) makes the assumption that the problem is convex in both \(f(x)\) and \(h(y)\) and that Slater’s conditions hold. Therefore, by Property 9.4 we know that strong duality holds and there exists a \(\lambda^* \in \mathbb{R}^K\) such that \(d^* = g(\lambda^*)\).

Since strong duality holds, then by Property 9.3, we know that for all \(x \in \Omega\) and \(z \in \Phi\), \(L(x, z; \lambda^*) \geq L(x^*, z^*; \lambda^*)\) where \(x^*\) and \(z^*\) are the assumed solutions to the constrained optimization problem.

Furthermore, since \((x^*, z^*)\) are the assumed solution, we know that \(Ax^* + Bz^* = b\). So therefore, we know that for all \(\lambda \in \mathbb{R}^K\) that

\[
L(x^*, z^*; \lambda^*) = L(x^*, z^*; \lambda) .
\]

With these two results, we have that

\[
L(x, z; \lambda^*) \geq L(x^*, z^*; \lambda^*) \geq L(x^*, z^*; \lambda) ,
\]

and the property is proved.

The following property then states that the ADMM algorithm converges for any convex optimization problem with the form of equation (9.65).
Property 9.7. Convergence of ADMM Algorithm - Consider a constrained convex optimization problem with the canonical form of equation (9.65). Then the ADMM iterations specified in Figure 9.8 converge for this problem in the following precise sense,

\[
\lim_{k \to \infty} \left\{ A\hat{x}^{(k)} + B\hat{z}^{(k)} - c \right\} = 0 \\
\lim_{k \to \infty} \left\{ f(\hat{x}^{(k)}) + h(\hat{z}^{(k)}) \right\} = p^* \\
\lim_{k \to \infty} u^{(k)} = \frac{\lambda^*}{a},
\]

where \(\hat{x}^{(k)}, \hat{z}^{(k)},\) and \(u^{(k)}\) denote the result of the \(k^{th}\) ADMM update.

Proof. The canonical problem of equation (9.65) makes the assumption that \(x^*\) and \(z^*\) exist that are solutions to the constrained optimization problem. Therefore, by Property 9.6 there must also exists a \(\lambda^* \in \mathbb{R}^K\) such that \((x^*, z^*, \lambda^*)\) is a saddle point to the constrained optimization problem of equation (9.65). In [14], it is shown that the stated convergence properties hold when \(f(x)\) and \(h(x)\) are closed proper convex functions, and \((x^*, z^*, \lambda^*)\) is a saddle point solution. So the proof is complete. \(\square\)
9.6 Chapter Problems

1. Let \( f : A \to \mathbb{R} \) is a convex function on the non-empty convex set \( A \subset \mathbb{R}^N \), and furthermore define the extended function
\[
g(x) = \begin{cases} 
  f(x) & \text{if } x \in A \\
  \infty & \text{otherwise} 
\end{cases}.
\]
Then prove that \( g : \mathbb{R}^N \to \mathbb{R} \) is a proper, closed, and convex function.

2. Consider the optimization problem
\[
\hat{x} = \arg \min_{x \in \mathbb{R}^N} \, f(x), \quad Cx - b \in \mathbb{R}^N
\]
where \( x \in \mathbb{R}^N, b \in \mathbb{R}^K \), and \( C \in \mathbb{R}^{K \times N} \). Then define the new vector
\[
z = \begin{bmatrix} x \\ y \end{bmatrix}
\]
where \( y \in \mathbb{R}^K \) and \( y = Cx - b \). Then show that there is an equivalent constrained optimization problem with the form
\[
\hat{x} = \arg \min_{z \in \mathbb{R}^N} \, f(z), \quad Bz = c
\]
for some appropriately chosen matrix \( B \) and vector \( c \). The variable \( y \) is sometimes referred to as a slack variable.

3. Show that for any function \( f : A \times B \to \mathbb{R} \) the following inequality holds.
\[
\inf_{y \in B} \sup_{x \in A} \, f(x, y) \geq \sup_{x \in A} \inf_{y \in B} \, f(x, y)
\]

4. Let \( f(x) \) be continuously differentiable on \( \Omega = \mathbb{R}^N \) and let \( x(\lambda) \), the solution to equation (9.9), also be continuously differentiable on \( \lambda \in \mathcal{D} \). Then show that the gradient of the dual function is given by
\[
\nabla g(\lambda) = Cx(\lambda) - b.
\]

5. Consider the constrained optimization problem of equation (9.6). Show that when the primal function \( f(x) \) is continuously differentiable on \( \mathbb{R}^N \), then \( \lambda^* \) achieves a maximum of the dual function \( g(\lambda) \) if and only if the constraint \( Cx(\lambda^*) - b = 0 \) is met and in this case \( p^* = d^* \).
6. In this problem, we prove that repeated application of a proximal mapping of a convex function $u(x)$ converges to its global minimum.\footnote{For simplicity, we will assume that $u(x)$ is continuously differentiable, but the same result holds without this assumption.} Let $u(x)$ be a continuously differentiable convex function on $\Omega \subset \mathbb{R}^N$, and define the proximal mapping function as

$$H(y) = \arg\min_{x \in \Omega} \left\{ ||y - x||^2 + u(x) \right\} . \quad (9.69)$$

Furthermore, define the function $f(x, y) = ||y - x||^2 + u(x)$.

a) Show that the following equality holds

$$\min_{y \in \mathbb{R}^N, x \in \Omega} f(x, y) = \min_{x \in \Omega} u(x) ,$$

and that the minimum is achieved by the same value of $x$ on both the left and right sides.

b) Use Theorem \[7.2.1] to show that the following recursion must converge to the minimum of $f(x, y)$.

$$x^{(k+1)} = \arg\min_{x \in \Omega} f(x, y^{(k)}) ,$$

$$y^{(k+1)} = \arg\min_{y \in \mathbb{R}^N} f(x^{(k+1)}, y) ,$$

c) Show that the recursion of part b) above has the form

$$y^{(k+1)} = H(y^{(k)}) ,$$

and that $\lim_{k \to \infty} y^{(k)} = x^*$ where $x^*$ achieves the minimum $u(x)$.

7. Let

$$L(x; a, \lambda) = f(x) + \lambda^t (Cx - b) + \frac{a}{2} ||Cx - b||^2$$

$$\tilde{L}(x; a, u) = f(x) + \frac{a}{2} ||Cx - b + u||^2 ,$$

where $f : \mathbb{R}^N \to \mathbb{R}$ and $\lambda = au$. Then show that $x^*$ achieves a local minimum of $L$ if and only if it achieves a local minimum of $\tilde{L}$.

8. Show that the shrinkage operator of equation (9.35) is the solution to the following minimization operation.

$$\arg\min_{x_i} \left\{ \frac{1}{2\sigma}(y_i - x_i)^2 + |x_i| \right\} = S_\sigma(y_i)$$
9. Consider the problem

\[ \hat{x} = \arg \min_{x \in \mathbb{R}^N} f(x), \]

where \( f : \mathbb{R}^N \to \mathbb{R} \) is a convex function. In order to remove the constraint, we may define the proper, closed, convex function

\[ g(x) = \begin{cases} 0 & \text{if } x \in \mathbb{R}^+^N \\ \infty & \text{if } x \not\in \mathbb{R}^+^N \end{cases}. \]

Then the minimum is given by the solution to the unconstrained optimization problem

\[ \hat{x} = \arg \min_{x \in \mathbb{R}^N} \{ f(x) + g(x) \}. \] (9.70)

Using this formulation, do the following.

a) Use variable splitting to derive a constrained optimization problem that is equivalent to equation (9.70).

b) Formulate the augmented Lagrangian for this constrained optimization problem and give the iterative algorithm for solving the augmented Lagrangian problem.

c) Use the ADMM approach to formulate an iterative algorithm for solving the augmented Lagrangian.

d) Simplify the expressions for the ADMM updates and give the general simplified ADMM algorithm for implementing positivity constraints in convex optimization problems.
Chapter 10

Plug-and-Play and Equilibrium Methods

Recent progress in machine learning and deep neural networks (DNN) has made it clear that these emerging methods have the capacity to represent very complex phenomena. This raises the question of how to best use them in model-based imaging applications. One obvious approach (which we will not treat here) is to simply learn the relationship between the data, $y$, and the unknown, $x$, using training data. However, in real applications this direct approach is often not desirable because one would like to incorporate detailed physical models, particularly for sensor models that we would rather not replace with machine learning algorithms. Also, direct learning methods tend to be one-size-fits all, without the modular structure that makes engineering designs flexible and maintainable.

In this chapter, we introduce the concepts of **Plug-and-Play** (PnP) models and their associated equilibrium methods. The fundamental goal of PnP is to allow the cost minimization models used up to this point in the text to be fused with action based models, such as DNNs, that are designed to generate outputs from inputs. The most direct example of this is the PnP prior in which the prior model is replaced by a general class of denoising operators. In order to put this new class of methods on a firm theoretical framework, we also introduce equilibrium methods to characterize their solution, and finish with the introduction of multi-agent consensus equilibrium methods (MACE).
10.1 Motivation and Intuition

As we have seen, inverse problems represent a very broad class of important problems in which the goal is to recover some unknown or latent quantity, $x$, from a set of measurements, $y$. The MAP estimate and regularized ML estimate are general methods for solving inverse problems that can be expressed in the form

$$x^* = \arg \min_x \{ f(x) + h(x) \},$$  \hspace{1cm} (10.1)

where we use the notation $x^*$ to emphasize its role as the solution to an optimization problem. For the MAP estimate, $f(x) = -\log p(y|x) + \text{const}$ and $h(x) = -\log p(x) + \text{const}$. More generally for regularized inversion, $f(x)$ can be any data fitting expression such as total squared error, and $h(x)$ can be any stabilizing function that penalizes irregular behavior in the solution.

![Figure 10.1: Inverse problems represent a broad category of important problems in which the goal is to estimate an unknown or latent quantity $x$ from available measurements $y$.](image)

This optimization framework for regularized inversion has served us well, but it has some severe limitations. First, it is not modular. So the optimization of both terms must be done together. This can be challenging in real applications for which the forward and prior model might represent 10s of thousands of lines of code! The splitting methods introduced in Chapter 9 provide some useful methods for breaking down large optimization problems into smaller manageable pieces, and we will build on these in this chapter. Second, the solution of equation (10.1) is purely based on the goals of minimizing the cost functions $f(x)$ and $h(x)$, but many powerful techniques are based on actions in the form of machine learning functions, $F(x)$, that take and input and transform it to some desired output.

Figure 10.2 illustrates graphically why there is a critical need for machine learning methods in the MAP inversion problems. The figure illustrates the
10.1 Motivation and Intuition

Figure 10.2: The MAP estimate or more generally the regularized inverse represents a balance between the goals of fitting the sensor data and being a plausible solution to the problem. For an image reconstruction problem using a linear sensor with AWGN, this can be viewed as finding a point in a high dimensional space that is close to both the sensor and image manifold.

Problem of reconstructing a MAP estimate that is as close as possible to both the thin sensor and prior manifolds. In this example, the sensor is assumed to be linear with additive white Gaussian noise (AWGN), so 

\[ f(x) = \frac{1}{2} \| y - Ax \|, \]

but the prior manifold is almost always very complex and nonlinear in structure. Intuitively, the prior thin manifold is always nonlinear because real images have enormously complex structure. Consequently, machine learning methods such as DNNs are serving an increasingly important role in accurately capturing the complex structure in the prior model’s thin manifold.

In the following section, we present the Plug-and-Play (PnP) algorithm for integrating an action base prior model with the optimization of a goal-based forward model. In later sections, we will show how the PnP method can be viewed as a special case of a larger class of equilibrium based solutions to regularize inverse problems. Equilibrium based problem formulations include traditional regularization approaches as a special case, but dramatically broaden the inverse problem formulation to include action-based operators that would otherwise not be possible in a traditional MAP framework.
10.1.1 The Plug-and-Play Algorithm and Agents

In Chapter 9 and Section 9.1, we introduced the concepts of the ADMM algorithm and variable splitting based on constrained optimization. In order to apply these techniques to the optimization of equation (10.1), we first need to introduce the proximal maps associated with the functions $f(x)$ and $h(x)$ defined in Section 9.4. These proximal maps are given by

$$F(x) = \arg \min_v \left\{ f(v) + \frac{1}{2\sigma^2} \|v - x\|^2 \right\}$$  

$$H(x) = \arg \min_v \left\{ h(v) + \frac{1}{2\sigma^2} \|v - x\|^2 \right\}$$

where $\sigma^2$ is a selectable parameter.

The proximal map is a very subtle but powerful idea because it provides a mathematical machine for converting goals to actions. For example, the function $f(x)$ represents a goal that we attempt to balance with other goals when computing the MAP estimate. However, the proximal map, $F(x)$, represents an action we take to achieve this goal because the application of $F(x)$ moves us closer to the minimum of $f(x)$.

Figure 10.3 graphically illustrates this difference between goals and actions. The function $f(x)$ in Fig. 10.3a represents a goal because the goal is to change $x$ in a way that reduces the value of the function. Alternatively, the operator $F(x)$ in Fig. 10.3b represents an action because when the function is applied to $x$ it creates the new value $F(x)$. In this context, the proximal map shown in Fig. 10.3c can be thought of as a method for converting a goal to an action. Intuitively, the proximal map is the resting position of a cart that is attached to a spring that is anchored at a point $x$. Gravity causes the cart to move down the function, but the spring restrains the cart and keeps it from moving to the bottom. So in this example, the goal, $f(x)$, creates an action, $F(x)$, that results in the movement of the cart.

As discussed in Section 9.1, we can use the proximal maps to split the optimization problem into two parts and iteratively solve the part separately using the ADMM algorithm. Figure 10.4 shows the PnP-ADMM algorithm.

---

1This physical analogy is not exact because in this case the spring extends diagonally rather than horizontally as the proximal map definition requires. But the emotional intuition you get by looking at the figure is correct.
that results from applying ADMM to our problem \cite{69, 64}. When $F(x)$ and $H(x)$ are the proximal maps from equation (10.3), then this is exactly the ADMM algorithm of the previous chapter. However, the power of PnP comes when we replace these functions $F(x)$ and $H(x)$ with more general actions. This is a huge conceptual step because when $F(x)$ and $H(x)$ are no longer proximal maps, then this algorithm is no longer solving an optimization problem. This is why we refer to this algorithm as “Plug-and-Play” ADMM, because we have simply plugged-in a different action where the proximal map should have been.

Figure 10.5 provides some intuition as to what the PnP algorithm is actually doing. Each iteration applies and action $F$ that moves the solution closer to the sensor manifold and an action $H$ that moves the solution closer to the data manifold. The hope is that these iterations will eventually converge to
Figure 10.4: Pseudo-code for the PnP-ADMM algorithm: When $F(x)$ and $H(x)$ are proximal maps for $f(x)$ and $h(x)$, then this results in the conventional MAP estimate. However, the PnP prior replaces $H(x)$ with an action designed or trained to remove AWGN with variance $\sigma^2$. The optimized denoiser then serves as a much better prior model, resulting in a much more effective reconstruction algorithm. The power of PnP is that it can use advanced machine learning methods, such as DNNs, that much more accurately model real images.

a solution that balances these two actions.

Figure 10.5: A graphical illustration of how PnP converges by iteratively searching for a solution that balances the two goals of being close to the sensor manifold and being close to the data manifold.

10.1.2 The Design of Plug-In Agents

OK, so this is all fine and well, but if we are not going to plug in the proximal map, then what new operator should we plug-in instead? This is a very important question which will lead us down some very interesting roads. Our
goal will be to replace the proximal maps in the ADMM algorithm with more general operations that represent more useful actions. From now on, we will refer to these operations as **agents** since we will see they have the anthropomorphic interpretation of imposing their will on the solution.

But for now, let us start by taking a closer look at the proximal map $H(x)$, which we have rewritten below for convenience.

$$H(x) = \arg \min_v \left\{ \frac{1}{2\sigma^2} \| x - v \|^2 + h(v) \right\}$$

(10.4)

In this form, we can recognize some familiar terms. The 1st term has the same mathematical form as the negative log likelihood one would get with a forward model that simply added white Gaussian noise to an image. So for example, imagine that we have a **virtual model** with the form

$$X_n = V_n + W_n,$$

where $V_n$ for $n = 1, \cdots, N$ is an image that is typical of what we expect in our reconstruction problem, and $W_k$ is synthetically generated AWGN with variance $\sigma^2$. Then in this case, the negative log likelihood is given by

$$-\log p(x|v) + \text{const} = \frac{1}{2\sigma^2} \| x - v \|^2 ,$$

which is exactly the same as the 1st term in the proximal map of equation (10.4). From this perspective, $H(x)$ has the interpretation of being an operator designed to compute the MAP estimate of $V_n$ given $X_n$ under the assumption the prior distribution of $V_n$ is given by $p(v) = \frac{1}{z} \exp \{-h(v)\}$.

Remember, $X_n$ and $V_n$ are not actually images in our original reconstruction problem. Instead, they are virtual images that we can imagine exist. However, just because they were born as virtual images, doesn’t mean we can’t generate them synthetically. To do this, we simply take examples of real images from our application, $V_n$, and we add Gaussian white noise with variance $\sigma^2$ to generate $X_n$. The resulting image pairs, $(V_n, X_n)$, can then be used for supervised training of $H$. From a machine learning perspective, the inference algorithm is trained using the MSE loss function
Figure 10.6: (a) A flow diagram of the process for supervised training of the denoising algorithm used as a PnP prior model. Notice that the agent, $H_{\theta}(X_n)$, is trained to remove additive white Gaussian noise from typical images. (b) A graphic interpretation of what the PnP prior model is trained to do. It maps any input image to a point closer to the prior manifold.

$$L(\theta) = \sum_n \| V_n - H_{\theta}(X_n) \|^2$$

Figure 10.6a graphically illustrates how the agent, $H_{\theta}(x)$, is trained to denoise $X_n$ in a manner that minimizes MSE loss. The resulting agent, $H_{\theta}(X_n)$, can then be thought of as an image denoiser since its output is then a MMSE estimate of the noiseless image, $V_k$. Figure 10.6b graphically illustrates the intuition behind this training procedure. The agent behaves similarly to a proximal map. Given an input image, $X_n$, the agent maps the image to a new point that is closer to the prior manifold.

---

2 Technically, this loss function results in the conditional mean rather than the MAP estimate, but in practice, this loss function is very effective for training the PnP denoiser.
10.1.3 Are All Agents Proximal Maps?

While this PnP-ADMM algorithm turns out to work amazingly well, it leaves open some serious questions. Why does it work? Does it converge? And if so, to what?

In order to understand the gravity of our questions, it’s useful to first make some important observations. First, we have already seen that the goal of $f(x)$ can be converted to the action $F(x)$ of its proximal map, but does the converse mapping always exist? The answer is a definitive “no”. This question was pioneered by Moreau in the 1960’s [48]. Based on this fundamental work, it turns out that for a continuously differentiable function $H(x)$ to be a proximal map, it must meet two conditions [64, 48]. First, it must be non-expansive. The second condition, which provides deeper insight, is that the gradient of $H(x)$ must be self-adjoint. Mathematically, this means

$$\nabla H(x)^t = \nabla H(x).$$

So what does this self-adjoint condition mean? Consider the example where $H(x) = Ax$ for some matrix $A$. In this case, $\nabla H(x) = A$, so the condition is that the matrix is symmetric, i.e., $A^t = A$. Generally speaking, this is a strong constraint that you would not necessarily expect to hold for many agents. In fact, this is analogous to a classic problem in electrodynamics of determining when a particular vector field can possibly be a valid electric field. Students of electrodynamics know that the answer is that such a potential field only exists when the electric field has “zero curl”, which is exactly a statement that the gradient of the field is self-adjoint.

The important point here is that many interesting actions, $H(x)$, may not have self-adjoint gradients, so they may not be proximal maps. Consequently, the much richer class of agents that are not limited to proximal maps offer the potential for much better reconstruction results. In other words, a DNN trained for optimal image denoising is not likely to be a proximal map, but it could serve as an excellent prior model in a PnP framework.

---

3Early on, an anonymous reviewer aptly renamed this algorithm “plug-and-pray”. 
10.2 Consensus Equilibrium for Two Models

Our goal in this section is to introduce a theoretical framework, which we call **consensus equilibrium**, that will allow us to fuse sets of heterogeneous models represented by both actions and goals. This will serve as a basis for integrating machine learning and physics-based models.

Looking more closely at the PnP-ADMM algorithm of Fig. 10.4, we can see that when the algorithm converges, it must be that $x = v$. Therefore, we know that the following two equations that must hold at convergence.

$$x^* = F(x^* - u) \quad (10.5)$$

$$x^* = H(x^* + u) \quad (10.6)$$

We refer to equations (10.5) and (10.6) as the **consensus equilibrium (CE) equations** for our inverse problem. The CE equation are the the defining criteria for the solution to our inverse problem $x^*$.

In the particular case when $F(x)$ and $H(x)$ are proximal maps corresponding to the convex functions $f(x)$ and $h(x)$, then the solution to the CE equations, $x^*$, is exactly the solution to the MAP optimization problem. This is true because we know that the ADMM algorithm converges to the global minimum of a convex problem. So in this case, we know that the actions of the CE equations are equivalent to the goals of the MAP estimate.

We can also make a more direct connection between the solution to the CE equations and the MAP estimate by taking the gradient of the CE equation with respect to $x^*$. In this case, we can easily show that

$$\nabla_x [f(x) + h(x)]|_{x=x^*} = 0 ,$$

where $x^*$ denotes the solution to the CE equations. So from this we can see that if the functions $f(x)$ and $h(x)$ are convex, then $x^*$ must also be the MAP estimate.\(^4\) (See Problem 11)

Figure 10.7a graphically illustrates why the solution to the MAP optimization problem of equation (10.1) is equivalent to the solution to the CE equations of (10.5) and (10.6) when the two agents are proximal maps. The figure shows the two functions $f(x)$ and $h(x)$ separately, each with their own

\(^4\) This proof depends on the function being continuously differentiable, but it is easily generalized using sub-gradients to the broader class of convex functions.
proximal map solution represented by the position of a cart, with each spring stretched by the same amount but in the opposite directions of $\pm u$. In this case, the minimum of the sum of the two functions occurs when the two carts fall to the same location.

More generally, Fig. 10.7b illustrates the solution to the CE equations when the agents are not proximal maps. In this case, the CE equations represent a force balance equilibrium in which agents are acting in opposing directions with equal action. However, this equilibrium does not result from the minimization of any underlying cost function because the agents are not proximal maps.
Again, we note that the actions used in the CE equations are much more general than the goals of the MAP estimate because, for example, we may choose a operation $H(x)$ that is not a proximal map. This mixing of actions and goals is the essence of the Plug-and-Play (PnP) method, and it allows us to, for example, fuse prior models represented by DNNs with goals represented by traditional sensor models. Whereas the convergence of the PnP algorithm was up to question, the CE equations give us a firm criteria for stating the solution to our problem.

### 10.2.1 Solving the CE Equations

Now the question becomes, “Do the CE equations of (10.5) and (10.6) have a solution? And if they do, then how can we compute it?”

In order to answer these questions, we will need to reorganize the CE equations into a much more elegant form. To this end, we start by introducing a change of variables of $w_1 = x - u$ and $w_2 = x + u$, so then the CE equations become

\[
F(w_1^*) = \frac{w_1^* + w_2^*}{2} \quad (10.7)
\]

\[
H(w_2^*) = \frac{w_1^* + w_2^*}{2},
\]

where now $u = \frac{w_2^* - w_1^*}{2}$ and $x^* = \frac{w_1^* + w_2^*}{2}$. Next, we define a new variable, $w$, by stacking the two states $w_1$ and $w_2$.

\[
w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}
\]

And we also define a corresponding stacked operator, $F(w)$, by stacking the two operators of $F$ and $H$.

\[
F(w) = \begin{bmatrix} F(w_1) \\ H(w_2) \end{bmatrix}
\]

Our next step introduces a subtly important new operator that will play a central role in answering our questions. The new operator, $G(w)$, computes the average of the two states in $w$ and is given by

\[
G(w) = \begin{bmatrix} \bar{w} \\ \bar{w} \end{bmatrix}
\]
where \( \bar{w} = \frac{w_1 + w_2}{2} \).

With this notation in hand, we can now express the CE equations of (10.7) and (10.8) in a much more compact and elegant form.

\[
F(w) = G(w) \quad (10.8)
\]

The averaging function, \( G \), is deceptively simple, but powerful. The key property of \( G \) is that if you apply it twice, you get the same result because the average of the average is the average. Symbolically, this means that \( GG = G \). This also means that \((2G - I)^2 = 4GG - 4G + I = I\), which in turn implies that \((2G - I)^{-1} = (2G - I)\). Using this fact, we get the following important result.

\[
Fw^* = Gw^* \quad (10.9)
\]

\[
(2F - I)w^* = (2G - I)w^* \quad (10.10)
\]

\[
(2G - I)(2F - I)w^* = w^* \quad (10.11)
\]

\[
Tw^* = w^* \quad (10.12)
\]

So we see that that the solutions to the CE equations of (10.8) are exactly the fixed points of the operator \( T = (2G - I)(2F - I) \).

This is important because there are many robust algorithms for solving for fixed points to choose from. In particular, we will use the Douglas-Rachford (DR) algorithm [?] with Mann iterations [?], which has the form

\[
w \leftarrow \rho w + (1 - \rho)Tw , \quad (10.13)
\]

where \( \rho \in (0, 1) \) is a user selectable parameter. Once the algorithm converges to a fixed point, \( w^* \), then the solution can be evaluated by averaging its two components given by \( x^* = \frac{w_1^* + w_2^*}{2} \). Importantly, the DR algorithm is guaranteed to converge when the operator \( T \) is non-expansive.

Algorithm [10.8] shows the pseudo-code implementation of the PnP-DR algorithm with the user-selectable parameter \( \rho \in (0, 1) \). Importantly, when \( \rho = 0.5 \) and \( F \) and \( H \) are proximal maps, then this PnP-DR algorithm can be shown to be exactly PnP with the consensus ADMM algorithm [64]. However, the PnP-DR algorithm as two important advantages as compared to consensus ADMM. The first and perhaps most important advantage is that PnP-DR has a well defined solution even when the operators \( F \) and
Plug-and-Play and Equilibrium Methods

**PnP-DR Algorithm**

Initialize Reconstruction: \( x^{(0)} \in \mathbb{R}^N \)

\[
\begin{align*}
w & \leftarrow \begin{bmatrix} x^{(0)} \\ x^{(0)} \end{bmatrix} \\
\text{Until Converged} \quad \{ \\
\quad w' & \leftarrow (2G-I)(2F-I)w \\
\quad w & \leftarrow \rho w + (1-\rho)w' , \\
\} \\
x^* & \leftarrow \frac{w_1 + w_2}{2} \\
\text{Return} (x^*)
\end{align*}
\]

Figure 10.8: Pseudo-Code for the PnP-DR Algorithm: This algorithm computes the solution to the CE equations for plug-in forward or prior agents \( F(x) \) and \( H(x) \) using the Douglas-Rachford Algorithm. The algorithm is provably convergent when \( T = (2G-I)(2F-I) \) is non-expansive.

\( H \) are not proximal maps. Remember, the ADMM algorithm is designed to minimize a cost function, but if the operators \( F \) and \( H \) are not proximal maps, then there is no underlying cost function to minimize! Nonetheless, even when \( F \) and \( H \) are not proximal maps, the PnP-DR algorithm is still designed to solve the CE equations.

The second advantage of PnP-DR is that by using larger values of \( \rho \) it is typically possible to get much more rapid convergence, thereby saving computation. Moreover, we can borrow a wide array of known existing results for the convergence of the DR algorithm in order to understand both existence of a CE solution, and the convergence of the PnP-DR algorithm. The most basic result is that the PnP-DR algorithm converges when \( T = (2G-I)(2F-I) \) is non-expansive. This is a far weaker condition than being a proximal map, which is required by ADMM.

### 10.3 Multi-Agent Consensus Equilibrium

We next consider the most general case in which we will solve for the consensus equilibrium solution when there are multiple heterogeneous agents. Some of these agents will have the forms of proximal maps corresponding to underlying cost functions or goals; and some of these agents will be directly
implemented as operations performing actions. We refer to this problem as multi-agent consensus equilibrium (MACE).

In order to best understand the MACE formulation, we start by assuming the special case that there are $K$ agents, each with a corresponding cost function, $f_k(x)$. In this special case, the complete cost function has the form

$$f(x) = \sum_{k=1}^{K} \mu_k f_k(x)$$  \hspace{1cm} (10.14)$$

where $\mu_k$ are weights such that $1 = \sum_{k=1}^{K} \mu_k$. Intuitively, the functions $f_k(x)$ may represent different soft or even hard constraints that one would like to incorporate into the solution of the inverse problem. The desired solution to the problem is then given by

$$x^* = \arg \min_x \left\{ \sum_{k=1}^{K} \mu_k f_k(x) \right\}.$$  \hspace{1cm} (10.15)$$

When each agent has a cost function, then each of the $K$ agents must also
have an associated proximal map given by

\begin{align*}
F_1(v) &= \arg\min_x \left\{ f_1(x) + \frac{1}{2\sigma^2} \|x - v\|^2 \right\} \\
&\vdots \\
F_K(v) &= \arg\min_x \left\{ f_K(x) + \frac{1}{2\sigma^2} \|x - v\|^2 \right\},
\end{align*}

where \( \sigma^2 \) is a selectable parameter that does not affect the final solution. Using the notation from the previous section, we have that the stacked state is given by

\[ w = \begin{bmatrix} w_1 \\ \vdots \\ w_K \end{bmatrix}, \]

and the stacked operator is given by

\[ F(w) = \begin{bmatrix} F_1(w_1) \\ \vdots \\ F_K(w_K) \end{bmatrix}. \]

Finally, we define a stacked weighted averaging operator as

\[ G_\mu(w) = \begin{bmatrix} \bar{w} \\ \vdots \\ \bar{w} \end{bmatrix}, \]

where \( \bar{w} = \sum_{k=1}^K \mu_k w_k \). Using this notation, the MACE equations can now be written in the compact form

\[ F(w^*) = G_\mu(w^*), \quad (10.16) \]

where the solution is now given by \( x^* = \sum_{k=1}^K \mu_k w_k^* \).

In the particular case when the agents are proximal maps, then by differentiating the MACE equations we can show that

\[ \nabla_x f(x)|_{x=x^*} = 0, \]

where \( x^* \) denotes the solution to the MACE equations. So if \( f(x) \) is convex, then the MACE solution must also be the solution to the minimization problem of equation (10.14).
However, the power of the MACE equations is when we use them with plug-in agents $F_k$ that are not proximal maps. In this case, the solution to the MACE equations does not correspond to the minimization of any underlying cost function, but it does represent an equilibrium that balances the agents actions.

Figure 10.9 provides a graphical interpretation of the MACE solution. Each agent is depicted as a action that moves the state from a starting point of $w_k$ to the common central point of $x^*$ that constitutes the MACE solution. We also have the additional constraint that $x^* = \sum_{k=1}^{K} \mu_k F_k(w_k)$.

The weights $\mu_k$ represent an important and very useful generalization of the CE method. By using these weights, it is very easy to adjust the relative strength of the different agents by simply adjusting the weighted average used in computing $G_{\mu}$. For example, this can be used to adjust the regularization by adjusting the relative weighting of the forward and prior agents. Notice that adjusting these weights does not require that one change the agents, $F_k$. This is important since if the agents are designed using machine learning methods, they may require computationally expensive retraining every time they are adjusted.

Regardless of the particular choice of weights, the averaging operator, $G_{\mu}$, still has the critical property that $G_{\mu} G_{\mu} = G_{\mu}$. Therefore, we once again know that the solution to the CE equations is given by

$$T w^* = w^* ,$$

were $T = (2G - I)(2F - I)$. Again, we can solve for the fixed point of $T$ using the DR algorithm with Mann iterations, and we know this algorithm will converge when $T$ is non-expansive.

Intuitively, you can imagine these functions as anthropomorphic agents, each of which as the assignment to move the solution in a way that solves “their” problem. Much like the endless number of vice presidents in an overly bureaucratic university, each MACE agent may attempt to enforce its preference, which might be in conflict with the goals or actions of other agents. The CE solution is the solution that achieves an equilibrium in the actions of all these agents.

---

5 At one point Purdue had as many as 16 vice presidents, each with an enormous and very expensive staff. However, the analogy is not perfect because, unlike MACE, Purdue vice presidents did not appear to achieve any useful results.
MACE-DR Algorithm

Initialize Reconstruction: $x^{(0)} \in \mathbb{R}^N$

$w \leftarrow \begin{bmatrix} x^{(0)} \\ x^{(0)} \end{bmatrix}$

Until Converged {

\begin{align*}
  x & \leftarrow F(w) \\
  z & \leftarrow G_{\mu}(2x - w) \\
  w & \leftarrow w + 2\rho(z - x),
\end{align*}

}\}

$x^* \leftarrow w_1$

Return($x^*$)

Figure 10.10: Pseudo-Code for the MACE-DR: This algorithm computes the solution to the CE equations for multiple plug-in agents $F_k(x)$ using the Douglas-Rachford algorithm. The algorithm is provably convergent when $T = (2G - I)(2F - I)$ is non-expansive.

Figure [10.10] shows the algorithm for multi-agent consensus equilibrium (MACE) reconstruction that implements the DR algorithm with Mann iterations. In the special case that the agents $F_k$ are proximal maps, then the algorithm converges to $x^*$, the solution of equation (10.15). However, the real power of this method is that it works with any combination of agents that are proximal maps (i.e., goal models) or not proximal maps (i.e., action models); and it converges, when $T$ is non-expansive, to a solution that balances all these agent’s forces.
10.4 Chapter Problems

1. Assume that $f(x)$ and $h(x)$ are continuously differentiable convex functions, and that

$$\hat{x} = \arg \min_x \{ f(x) + h(x) \} .$$

Further assume that

$$x^* = F(x^* - u)$$
$$x^* = H(x^* + u) ,$$

(10.17)

where $F(x)$ and $H(x)$ are the proximal maps for $f(x)$ and $h(x)$ respectively, using the same parameter $\sigma^2$. Then prove that $x^* = \hat{x}$. 
Chapter 11

Model Parameter Estimation

The focus of the previous chapters has been on the estimation of an unknown image $X$ from measurements $Y$. However, in real applications there are often a variety of model parameters that must also be estimated. So for example, in an imaging system there are typically calibration parameters that control focus or data fidelity that must be accurately estimated in order to achieve high quality results. While in principle it might seem that these parameters can be determined in advance through a calibration or training procedure, in practice this is often impractical, particular when these parameters tend to drift with time.

A huge practical advantage of model-based methods is that they provide a principled framework to estimate model parameters as part of the reconstruction process. This avoids the need to do difficult or sometimes impossible calibration measurements, and it makes the reconstruction algorithms adaptive to variations in both image statistics and system calibration. The following chapter introduces a variety of methods for estimation of system and prior parameters, and also introduces the challenging problem of estimating discrete model order.

11.1 Parameter Estimation Framework

In practical problems, there are often unknown parameters for both the forward and prior models. So for example, the amplitude of measurement noise or the point-spread function of a sensor are quantities that can be modeled as unknown parameters. In addition, parameters such as the scaling
Figure 11.1: A very powerful aspect of model-based methods is that they provide a framework for the estimation of unknown hyper-parameters as part of the inversion process. These hyper-parameters can represent either unknown parameters of the system model, $\phi$, or unknown parameters of the prior model, $\theta$.

Parameter in an MRF prior model, as discussed in Section 6.4, is a classic example of an unknown parameter that is essential to know in order to obtain an accurate MAP estimate. These extra parameters, collectively known as hyper-parameters or perhaps more classically as nuisance parameters, are often not of direct interest, but are required in order to make good estimates of $X$.

Figure 11.1 illustrates our basic approach. The variable $\phi$ will be used to represent unknown parameters of the imaging system. So for example, parameters dealing with the system PSF, noise calibration, or data offsets might be parameterized by $\phi$. The variable $\theta$ will be used to represent unknown parameters of the prior model. This will include the all important regularization parameters that must typically be estimated in order to get a estimate of $X$ that is neither over nor under smoothed.

One relatively simple and intuitive approach to estimating $\phi$ and $\theta$ is to jointly minimize the MAP cost function with respect to all the unknown quantities \[^{[47]}\]. This results in

$$ (\hat{x}, \hat{\theta}, \hat{\phi}) = \arg \min_{x,\theta,\phi} \{- \log p(y|x, \phi) - \log p(x|\theta)\} . $$

We will refer to this as the joint-MAP estimate of $(x, \theta, \phi)$ because it can be viewed as a MAP estimate using uniform priors on the parameters \[^{[47]}\]. This joint estimate has the intuitive appeal of seeming to be a joint MAP estimate of $X$ and a ML estimate of $(\theta, \phi)$; however, in fact it is neither, and therefore it inherits the properties of neither. In fact, we will see that in many typical cases, this estimator is not even consistent. This is perhaps surprising since

\[^{1}\text{Such a prior is, of course, improper, but nonetheless can be convenient to assume in practice.}\]
11.2 Noise Variance Estimation

Typically, the joint-MAP estimation of system parameters, $\phi$, works well, and is easy to implement. In practice, $\phi$ can represent a wide variety of unknown physical parameters in sensing systems, so the ability to automatically estimate these parameters as part of the MAP estimation procedure is a huge advantage.

In order to illustrate how this is done, let’s consider a simple and useful case of estimating the unknown noise gain in a measurement system. So consider the case where

$$Y = AX + W$$

where $W$ is distributed as $N(0, \sigma^2\Lambda^{-1})$, so the inverse covariance of the additive noise is given by $\frac{1}{\sigma^2}\Lambda$ where $\sigma^2$ is an unknown scaling constant. This situation occurs commonly when the relative noise of measurements is known,
but the overall gain of the noise is unknown. For example, it is common in imaging systems for the conditional distribution of $Y$ given $X$ to have a Poisson distribution with mean $AX$. The Poisson noise model often results from the “shot” or photon counting noise when quanta of energy are detected in physical sensors. In this case, the noise can be approximated by independent additive Gaussian noise with mean zero and variance proportional to the amplitude of the signal $Y$.\footnote{59}

So consider a system where each measurement $Y_i$ is formed by amplifying a Poisson random variable $\lambda_i$ by an unknown gain constant $\alpha$. In practice, this happens in an “integrating detector” system in which the underlying signal, $\lambda_i$, is the Poisson random variable generated by some kind of photon counting process. An analog amplifier with a gain of $\alpha$ then produces the signal $Y_i = \alpha \lambda_i$ that is converted to digital form and processed on a computer. Using this model, it can be easily shown that

$$\text{Var}[Y_i | X] = \alpha \mathbb{E}[Y_i | X]$$

where $\alpha$ is an unknown gain constant that affects the overall level of noise variance.

A common assumption in this case is to assume that $Y_i$ are independent Gaussian random variables. So then the vector is distributed as $Y \sim N(AX, \alpha \Lambda^{-1})$ where $\Lambda$ is a diagonal matrix with diagonal entries of

$$\alpha \Lambda^{-1}_{i,i} \propto \mathbb{E}[Y_i | X] .$$

However, even though $\Lambda$ is known, it is often the case that the gain, $\alpha$ must still be estimated. In order to do this, we first write out the MAP cost function.

$$f(x, \alpha; y) = -\log p(y | x, \alpha) - \log p(x)$$

$$= \frac{1}{2\alpha} ||y - Ax||_\Lambda^2 + \frac{M}{2} \log (\alpha) - \frac{1}{2} \log (|\Lambda|) - \log p(x) (11.3)$$

Interestingly, this function can be minimized with respect to $\alpha$ in closed form to yield

$$\hat{\alpha} = \arg \min_{\alpha} f(x, \alpha; y)$$

$$= \frac{1}{M} ||y - Ax||_\Lambda^2 .$$
11.2 Noise Variance Estimation

Noise Variance Estimation using Alternating Minimization:

Initialize $\hat{x}$ and $\hat{\sigma}^2$

For $K$ iterations {

//Compute the MAP estimate using $\sigma^2 = \hat{\sigma}^2$.

$\hat{x} \leftarrow \arg \min_{x \in \Omega} \left\{ \frac{1}{2\hat{\sigma}^2} ||y - Ax||_\Lambda^2 - \log p(x) \right\}$

//Compute ML estimate of $\sigma^2$ using $x = \hat{x}$.

$\hat{\sigma} \leftarrow \frac{1}{M} ||y - A\hat{x}||_\Lambda^2$.

}

Figure 11.2: Pseudocode for alternating minimization of the MAP cost function with respect to both $x$ and $\sigma^2$. Each step monotonically reduces the cost of equation (11.4).

Then substituting this estimate of the noise variance into the MAP cost function, results in the following expression.

$$f(x, \hat{\sigma}; y) = \frac{M}{2} (1 - \log M) + \frac{M}{2} \log \left( ||y - Ax||_\Lambda^2 \right) - \frac{1}{2} \log (|\Lambda|) - \log p(x)$$

So from this, we can see that joint-MAP estimation will result in an estimate of $\hat{x}$ that minimizes a modified cost function given by

$$\tilde{f}(x; y) = \frac{M}{2} \log \left( ||y - Ax||_\Lambda^2 \right) - \log p(x) \quad (11.4)$$

Notice that in this form, we have removed any dependency on the unknown noise gain, $\alpha$, by replacing the norm squared term with its log. Minimization of this modified cost function results in the joint-MAP estimate of $X$ given by

$$\hat{x} = \arg \min_{x \in \Omega} \left\{ \frac{M}{2} \log \left( ||y - Ax||_\Lambda^2 \right) - \log p(x) \right\}.$$ 

This modified cost function, $\tilde{f}(x; y)$, is useful for tracking convergence of optimization algorithms, but in practice, direct minimization of this cost function is usually not the best approach. Usually, the best approach to minimizing equation (11.4) is to alternately minimize the joint MAP cost function of equation (11.3) with respect to $x$ and $\alpha$.

Figure 11.2 provides a pseudocode example of how this alternating minimization approach is performed. First the MAP estimate of $x$ is computed
under the assumption that \( \alpha = \hat{\alpha} \), and then the ML estimate of \( \alpha \) is computed using the estimate of \( x = \hat{x} \). Each step of this procedure monotonically decreases the value of the modified MAP cost function in equation (11.4). So the process can be repeated until convergence is achieved.

11.3 Scale Parameter Selection and Estimation

Perhaps the most important parameter to be selected in MAP estimation is the scale parameter of the prior distribution, \( \sigma_x \). Section 6.4 presented very tractable methods for computing the ML estimate of \( \sigma_x \) given observations of the MRF, \( X \). However, in the problem of MAP estimation, \( X \), is unknown; so it is not observed directly. We will see that one solution to this dilemma of incomplete data is to employ the techniques of the EM algorithm and stochastic sampling, to be discussed in Chapters 12 and 15. However, even in the best of situations, these approaches can be very computational expensive, so alternative methods are usually needed in practice.

In fact, scale parameter estimation has been studied for many years and is still a problem of great interest to the research community; so a consensus on the best approach has yet to emerge. Moreover, while very effective techniques do exist, they tend to be tuned to a specific class of applications and/or be very computational expensive. One issue that makes the selection of \( \sigma_x \) so difficult is that its actual value can vary dramatically based on the units and scale of the quantity \( X \). So if individual pixels in \( X \) represent interplanetary distances, and the units are meters, then we might expect an appropriate value for \( \sigma_x \) to be quite large. However, if the quantities being measured are molecular distances, but with the same units of meters, then \( \sigma_x \) will be many orders of magnitude smaller. Another challenge is that the “best” value of \( \sigma_x \) typically depends on the specific use of the image \( X \) being estimated. In some applications, it maybe more important to reduce noise in \( X \) at the cost of additional smoothing, but in other applications, it might be more important to retain detail. In practice, what is typically needed is a unitless quantity to adjust in a limited range that allows resolution and noise to be controlled about a reasonable nominal value.

One approach to the selection of \( \sigma_x \) is to use the joint-MAP estimator of equation (11.1). Assuming a Gibbs distribution with energy function \( u(x/\sigma_x) \),
then the prior distribution is given by
\[ p(x) = \frac{1}{z_0 \sigma_x^N} \exp \{-u(x/\sigma_x)\} \quad (11.5) \]
where \( N \) is the dimension of \( x \). So in this case, the MAP cost function is
given by
\[ f(x, \sigma_x; y) = -\log p(y|x) + u(x/\sigma_x) + N \log(\sigma_x) \quad (11.6) \]
This function can be minimized by alternately minimizing with respect to \( x \) and \( \sigma_x \). Minimization with respect to \( x \) is performed using the MAP estimation techniques we have discuss throughout this chapter, and the minimization of with respect to \( \sigma_x \) can be done by numerically rooting equation (6.22) of Section 6.4.

However, in practice, there is a serious problem with estimation of \( \sigma_x \) through the joint minimization of (11.6). The problem is that algorithms for minimization of (11.6) typically converge to the global minimum at \( \sigma_x = 0 \)!

In fact, after a bit of thought, it is clear that when \( x = 0 \), then \( \lim_{\sigma_x \to 0} f(0, \sigma_x; y) = -\infty \), so that \( \sigma_x = 0 \) is always a global minimum.\(^2\)

A somewhat ad-hoc but useful approach to fixing this problem is to introduce a unitless adjustable sharpening factor, \( \gamma > 1 \), which a user can tune to achieve the desired level of regularization. This sharpening parameter can be adjusted without knowing the scale of \( X \), so as to achieve the desired level of sharpness in the reconstruction. Using this procedure, the modified cost function is given by
\[ f(x, \sigma_x; y, \gamma) = -\log p(y|x) + u(x/\sigma_x) + \frac{N}{\gamma} \log(\sigma_x) \quad (11.7) \]
where \( \gamma \) is inserted as a way of attenuating the attraction of \( \sigma_x \) towards 0. So using the methodology described in Section 6.4, the update of \( \hat{\sigma}_x \) is computed by minimizing the cost function of equation (11.7), and this is done by rooting the following equation.
\[ \frac{\gamma \sigma_x}{N} \frac{d}{d\sigma_x} u(x/\sigma_x) \bigg|_{\sigma_x = \hat{\sigma}_x} = -1 \quad (11.8) \]

\(^2\) The existence of a minimum at \( \sigma_x = 0 \) is not necessarily so bad since for many problems, such as for example the estimation of the parameters of a Gaussian mixture model (as described in Section 12.2), there is an undesirable global minimum when the variance parameter is zero and a cluster contains only a single sample. However, the estimation of \( \sigma_x \) in MAP estimation is of greater concern because, in practice, there is often not even a local minimum to the cost function for which \( \sigma_x \) is bounded away from zero.
We can get more intuition into this procedure by considering the special case when the prior is a MRF using a GGMRF potential function so that
\[
u(x) = \frac{1}{p\sigma^p_x} \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} |x_s - x_r|^p.
\] (11.9)

In this case, the cost function of (11.7) can be minimized by alternating minimization with respect to the quantities \(x\) and \(\sigma_x\). This is done by repeating the following two update procedures.

\[
\hat{x} \leftarrow \arg \min_{x \in \Omega} \left\{ -\log p(y|x) + \frac{1}{p\sigma^p_x} \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} |x_s - x_r|^p \right\}
\] (11.10)

\[
\hat{\sigma}^p_x \leftarrow \frac{\gamma}{N} \sum_{\{s,r\} \in \mathcal{P}} b_{s,r} |\hat{x}_s - \hat{x}_r|^p.
\] (11.11)

So notice that in this procedure the estimated value of \(\sigma_x\) is increased by a factor of \(\gamma\) with each iteration. This gain factor reduces the amount of regularization and helps to compensate for the systematic underestimation of \(\sigma_x\) that tends to occur in a joint-MAP estimation procedure of this kind. The sharpening factor, \(\gamma > 1\), can also be used to adjust the amount of regularization to suit the application. So a large value of \(\gamma\) will result in a sharper image with less regularization.

Another problem that commonly arises is the need to adjust the scale parameter, \(\sigma_x\), as the sampling resolution of an image is changed. So for example, consider the case when \(X_s\) is an image on a discrete rectangular lattice with sample spacing \(h\) in each dimension. This occurs when \(X_s\) is a discretely sampled representation of an underlying continuous space function. More specifically, the discrete function \(X_s\) is often a representation of a continuous function \(f: \mathbb{R}^d \to \mathbb{R}\), where \(f(hs) = X_s\). In other words, \(f(u)\) is then a continuous function of \(u \in \mathbb{R}^d\), and \(X_s\) is a sampled version of \(f(u)\) for \(u = hs\).

So in this type of application, our underlying objective is to recover the continuous function, \(f(u)\), and we may want or need to change the choice of \(h\) used in the MAP reconstruction problem. For example, sometimes multi-resolution methods are used to solve the MAP reconstruction problems at different resolutions. For each of these resolutions, the value of \(h\) will change,
but we would like the underlying estimation problem for \( f(u) \) to remain the same. In order to do this, we need to scale the prior term of the MAP cost function so that it remains approximately constant as \( h \) changes.

We can make the MAP estimate approximately invariant to the sample spacing, \( h \), by scaling the energy function of the prior so that it has the following normalized form,

\[
u(x) = h^d \sum_{\{s,r\} \in \mathcal{P}} b_{s-r} \rho \left( \frac{x_s - x_r}{h} \right) . \tag{11.12}
\]

where we assume that \( s \) takes values on a \( d \)-dimensional rectangular lattice and \( b_{s-r} \) is only a function of the vector difference between the indices \( s \) and \( r \). Then the following series of approximate equalities illustrate how as \( h \) becomes small, this energy function converges toward the Riemann sum of a continuous integral.

\[
u(x) = \sum_{s \in S} \sum_{r \in W} b_r \rho \left( \frac{x_{s+r} - x_s}{h} \right)
\approx \frac{1}{2} \sum_{s \in S} b_r \sum_{r \in W} h^d \rho (r^t \nabla f(\hs))
\approx \frac{1}{2} \sum_{r \in W} b_r \int_{\mathbb{R}^d} \rho (r^t \nabla f(u)) \, du ,
\]

where the fourth line uses the Riemann sum approximation of an integral, and the third line uses the fact that

\[
\frac{x_{s+r} - x_s}{h} = \frac{f(hs + hr) - f(hs)}{h} \\
\approx \frac{hr^t \nabla f(\hs)}{h} \\
= r^t \nabla f(\hs) .
\]

From this result, we can see that the normalized energy function of equation (11.12) is approximately invariant to the choice of \( h \).
11.4 Order Identification and Model Selection
11.5 Chapter Problems

1. The following problems deal with joint estimation of $x$ and $\sigma^2_x$.
   
   (a) Derive an expression, analogous to (11.2) for the joint estimation of $x$ and $\sigma^2_x$.
   
   (b) Would this be a good estimator? Why or why not?
Chapter 12

The Expectation-Maximization (EM) Algorithm

Hopefully, the previous chapters have convinced you that the methods of model-based image processing are quite promising. However, one crucial question that arises is how should the model be chosen? Clearly, a poor choice of models will produce bad results, so this is a critical question.

One solution to this problem is to select a broad family of models that are controlled by a scalar or vector parameter. So for example, the prior model for an image may be controlled by a parameter which determines properties such as variation or edginess. These model parameters can then be estimated along with the unknown image. If the family of models is sufficiently broad, and the parameter estimation is sufficiently robust, then it should be possible to select a model from the family that accurately describes the data.

However, there is still a catch with this approach: The undistorted image is typically not available. In fact, if it were available, then we would not need to solve the image restoration or reconstruction problem in the first place. This means that the parameters that specify a particular prior model must be estimated from indirect measurements of the corrupted, distorted, or transformed image.

In fact, the problem of estimating parameters from indirect or incomplete observations is a recurring theme in model-based image processing. Fortunately, there are some powerful tools that can be used to address this problem. Among these tools, the expectation-maximization (EM) algorithm is perhaps the most widely used. At its core, the EM algorithm provides a two-step process for parameter estimation. In the first step, one conjectures
values for the missing data and uses them to calculate an ML estimate of the unknown parameters (M-step); and in the second step, the conjectured values are updated using the estimated parameter values (E-step). The two-step process is then repeated, with the hope of convergence.

This cyclic process is quite intuitive, but will it work? In fact, naive implementation of this approach leads to very bad estimates. However, the EM algorithm provides a formal framework for this iterative heuristic, which ensures well defined properties of convergence and good behavior.

The following Sections 12.1 and 12.2 first provide some informal motivation for the EM algorithm and its primary application in data clustering with Gaussian mixtures. Section 12.3 next builds the theoretic foundation and general form of the EM algorithm as it applies to many applications. Later sections develop the tools and motivate its use in an array of signal processing and imaging applications.

12.1 Motivation and Intuition

Imagine the following problem. You have measured the height of each plant in a garden. There are $N$ plants, and you know that some have been regularly fertilized, and the remainder have not been fertilized at all. Unfortunately, the fertilizing records were lost. Your measurements, $Y_n$, of the plant height for the $n^{th}$ plant could have been modeled as Gaussian with mean $\mu$ and variance $\sigma^2$, if they had all been treated equally; but since they have not, the fertilized plants will, on average, be taller.

We can model this unknown treatment of the plants by a random variable

$$X_n = \begin{cases} 
1 & \text{if plant has been fertilized} \\
0 & \text{if plant has not been fertilized}
\end{cases}.$$ 

Now if $X_n$ were known, then the distribution of plant heights for each category could be assumed to be Gaussian, but with different means, and perhaps the same variance $\sigma^2$. This implies that the conditional distribution of $Y_n$ given
Figure 12.1: Example of the distribution we might expect in plant height for the two populations. Notice that the distributions for fertilized and unfertilized plants are different, with the fertilized plants having a larger mean height. However, if the labels of the plants are unknown, then the distribution appears to have two modes, as shown in the second plot. Notice that naive estimates of the mean, are systematically offset from the true values.

$$X_n$$ will be Gaussian,

$$p(y_n|x_n) = \begin{cases} 
\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2} (y_n - \mu_1)^2 \right\} & \text{for } x_n = 1 \\
\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2} (y_n - \mu_0)^2 \right\} & \text{for } x_n = 0 ,
\end{cases}$$

with $$\mu_0$$ and $$\mu_1$$ denoting the means of the two different classes.

It might also be reasonable to assume the class labels of the plants, $$X_n$$, are i.i.d. Bernoulli random variables with

$$p(x_n) = \begin{cases} 
\pi_1 & \text{for } x_n = 1 \\
\pi_0 = 1 - \pi_1 & \text{for } x_n = 0 ,
\end{cases}$$

where $$\pi_i$$ parameterizes the Bernoulli distribution.

This type of stochastic process is sometimes referred to as a **doubly stochastic process** due to its hierarchical structure, with the distribution of $$Y_n$$ being dependent on the value of the random variable $$X_n$$. The parameters of this doubly stochastic process, $$(X_n, Y_n)$$, are then given by $$\theta = [\pi_0, \mu_0, \pi_1, \mu_1]$$ where $$\pi_0 + \pi_1 = 1$$. 
The question arises of how to estimate the parameter $\theta$ of this distribution? This is an important practical problem because we may want to measure the effect of fertilization on plant growth, so we would like to know how much $\mu_0$ and $\mu_1$ differ. Figure 12.1 illustrates the situation. Notice that the two populations create two modes in the distribution of plant height. In order to estimate the mean of each mode, it seems that we would need to know $X_n$, the label of each plant. However, casual inspection of the distribution of Figure 12.1 suggests that one might be able to estimate the unknown means, $\mu_0$ and $\mu_1$, by looking at the combined distribution of the two populations.

One possibility for estimating $\mu_0$ and $\mu_1$ is to first estimate the labels $X_n$. This can be done by applying a threshold at the valley between the two modes, and assigning a class of “fertilized” ($X_n = 1$) or “unfertilized” ($X_n = 0$) to each plant.

$$\hat{X}_n = \begin{cases} 1 & Y_n \geq \text{threshold} \\ 0 & Y_n < \text{threshold} \end{cases}$$ (12.1)

The result of this classification can be compactly represented by the two sets $S_0 = \{n : \hat{X}_n = 0\}$ and $S_1 = \{n : \hat{X}_n = 1\}$. Using this notation, we can estimate the two unknown means as

$$\hat{\mu}_0 = \frac{1}{|S_0|} \sum_{n \in S_0} Y_n$$

$$\hat{\mu}_1 = \frac{1}{|S_1|} \sum_{n \in S_1} Y_n,$$

where $|S_0|$ and $|S_1|$ denote the number of plants that have been classified as unfertilized and fertilized respectively.

While this is an intuitively appealing approach, it has a very serious flaw. Since we have separated the two groups by their height, it is inevitable that we will measure a larger value for $\hat{\mu}_1$ than we should (and also a smaller value of $\hat{\mu}_0$ than we should). In fact, even when the two means are quite different the resulting estimates of $\hat{\mu}_0$ and $\hat{\mu}_1$ will be systematically shifted no matter how many plants, $N$, are measured. This is much worse than simply being biased. These estimates are inconsistent because the estimated values do not converge to the true parameters as $N \to \infty$.

Another approach to solving this estimation problem is to attempt to directly estimate the value of the parameter vector $\theta = [\pi_0, \mu_0, \pi_1, \mu_1]$ from
12.2 Gaussian Mixture Distributions

Imagine a generalization of our example of the previous section in which the random variables \( \{Y_n\}_{n=1}^{N} \) are assumed to be conditionally Gaussian and independent given the class labels \( \{X_n\}_{n=1}^{N} \). In order to be a bit more general, we will also assume that each label takes on \( M \) possible values. So that \( X_n \in \{0, \cdots, M - 1\} \).

Figure [12.2] graphically illustrates the behavior of the model. Each observation can come from one of \( M \) possible Gaussian distributions, and the selection of the specific Gaussian distribution is made using a switch controlled by the class label \( X_n \). So for the \( m^{th} \) Gaussian distribution, the mean
and variance are denoted by $\mu_m$ and $\sigma^2_m$ and the distribution is denoted by $N(\mu_m, \sigma^2_m)$. Using this notation, if the control variable takes on the value $m = x_n$, then $Y_n$ is conditionally Gaussian with mean $\mu_{x_n}$ and variance $\sigma^2_{x_n}$.

With this model and notation, the conditional distribution of $Y_n$ given $X_n$ can be explicitly written as

$$p(y_n|x_n) = \frac{1}{\sqrt{2\pi\sigma^2_{x_n}}} \exp \left\{ -\frac{1}{2\sigma^2_{x_n}} (y_n - \mu_{x_n})^2 \right\},$$

where $\mu_{x_n}$ and $\sigma^2_{x_n}$ are the mean and variance of the sample when $x_n$ is the label. From this, we can calculate the marginal distribution of $y_n$ given by

$$p(y_n) = \sum_{m=0}^{M-1} p(y_n|m)\pi_m$$

$$= \sum_{m=0}^{M-1} \frac{\pi_m}{\sqrt{2\pi\sigma^2_m}} \exp \left\{ -\frac{1}{2\sigma^2_m} (y_n - \mu_m)^2 \right\},$$

(12.2)

where $\pi_m = P\{X_n = m\}$. If we further assume that the samples, $Y_n$, are i.i.d., then the distribution of the entire sequence, $\{Y_n\}_{n=1}^N$, can be written as

$$p(y) = \prod_{n=1}^N \sum_{m=0}^{M-1} \frac{\pi_m}{\sqrt{2\pi\sigma^2_m}} \exp \left\{ -\frac{1}{2\sigma^2_m} (y_n - \mu_m)^2 \right\}.$$  

(12.3)

A distribution with this form of (12.2) is known as a **Gaussian mixture distribution** with parameter vector $\theta = [\pi_0, \mu_0, \sigma^2_0, \cdots, \pi_{M-1}, \mu_{M-1}, \sigma^2_{M-1}]$. 

---

**Figure 12.2**: This figure graphically illustrates the creation of a random variable $Y_n$ with a Gaussian mixture distribution. The output is selected among $M$ possible Gaussian distributions using a switch. The switch is controlled using an independent random variable, $X_n$, that represents the class label.
Gaussian mixtures are very useful because, for sufficiently large $M$, they can approximate almost any distribution. This is because, intuitively, any density function can be approximated by a weighted sum of Gaussian density functions.

So the next question is, how can we estimate the parameter $\theta$ from the observations of $Y_n$? For now, we will make the simplifying assumption that we know both the values $Y_n$ and their associated labels $X_n$. In order to calculate the ML estimate, we first compute $N_m$, the number of labels taking on class $m$ given by

$$N_m = \sum_{n=1}^{N} \delta(X_n - m),$$

where $\delta(k)$ is 1 when $k = 0$ and zero otherwise. From this, we can compute the ML estimate of $\pi_m$ as the fraction of labels taking on the class $m$.

$$\hat{\pi}_m = \frac{N_m}{N}$$

The ML estimate of the mean for each class is then computed by only averaging the values of $Y_n$ with the class label $X_n = m$.

$$\hat{\mu}_m = \frac{1}{N_m} \sum_{n=1}^{N} Y_n \delta(X_n - m)$$

Finally, the ML estimate of the variance for each class is computed by only summing over terms with the class label $X_n = m$.

$$\hat{\sigma}_m^2 = \frac{1}{N_m} \sum_{n=1}^{N} (Y_n - \hat{\mu}_m)^2 \delta(X_n - m)$$

Of course, for our problem the class labels, $X_n$, are not known; so the problem remains of how we are going to calculate the ML estimate of $\theta$ from only the incomplete data $Y$. The EM algorithm provides a solution in the form of an algorithm for computing the ML parameter estimate that does not require the missing data. In essence, this is done by replacing the unknown labels, $X_n$, with their expected values. However, in order to fully understand this, we must first introduce the underlying theory of the method.
12.3 Theoretical Foundation of the EM Algorithm

In this section, we derive the mathematical relationships that serve as a foundation for the EM algorithm. While these relationships may seem abstract, they are relatively simple to derive, and are very powerful, so the reader is strongly encouraged to take the time to understand them. An early, succinct, general, and clear presentation of these results can be found in [3, 2].

The EM algorithm is based on two non-obvious mathematical insights. The first insight is that one can separate the log likelihood into the sum of two functions. In particular, the log likelihood can be expressed as

$$\log p(y|\theta) = Q(\theta; \theta') + H(\theta; \theta')$$  \hfill (12.4)

where $\theta'$ is any value of the parameter, and the functions $Q$ and $H$ are defined as

$$Q(\theta; \theta') \triangleq \mathbb{E}[\log p(y, X|\theta)|Y = y, \theta']$$  \hfill (12.5)

$$H(\theta; \theta') \triangleq -\mathbb{E}[\log p(X|y, \theta)|Y = y, \theta'].$$  \hfill (12.6)

where $p(y, x|\theta)$ is assumed to be strictly positive density function. In order to prove that the result of (12.4) holds, we first observe the following two equalities.

$$p(y|\theta) = \frac{p(y, x|\theta)}{p(x|y, \theta)} = \frac{p(y, X|\theta)}{p(X|y, \theta)}$$

The first equality is simply a result of conditional probability that must hold for all values of $x$. Since this equality holds for all $x$, it must in particular hold if we substitute in the random variable $X$. Using these identities, the following sequence of equalities then proves the desired result.

$$\log p(y|\theta) = \mathbb{E}[\log p(y|\theta)|Y = y, \theta']$$

$$= \mathbb{E}\left[ \log \left\{ \frac{p(y, X|\theta)}{p(X|y, \theta)} \right\} \bigg| Y = y, \theta' \right]$$

$$= \mathbb{E}[\log p(y, X|\theta)|Y = y, \theta'] - \mathbb{E}[\log p(X|y, \theta)|Y = y, \theta']$$

$$= Q(\theta; \theta') + H(\theta; \theta')$$

The second insight is that the function $H(\theta; \theta')$ takes on its minimum value when $\theta = \theta'$. More precisely, for all $\theta, \theta' \in \Omega$, we have that

$$H(\theta; \theta') \geq H(\theta'; \theta').$$  \hfill (12.7)
To see that this is true, we have the following set of inequalities

\[
0 = \log \left\{ \int p(x|y, \theta) \, dx \right\} = \log \left\{ \int \frac{p(x|y, \theta)}{p(x|y, \theta')} p(x|y, \theta') \, dx \right\} \geq \int \log \left\{ \frac{p(x|y, \theta)}{p(x|y, \theta')} \right\} p(x|y, \theta') \, dx = \int \log p(x|y, \theta) p(x|y, \theta') \, dx - \int \log p(x|y, \theta') p(x|y, \theta') \, dx = -H(\theta; \theta') + H(\theta'; \theta'),
\]

where the third line depends on the famous Jensen’s inequality which states that a concave function of the expectation is greater than or equal to the expectation of the concave function.

The two results of equations (12.4) and (12.7) now admit the foundation of the EM algorithm. If we adjust the parameter from an initial value of \( \theta' \) to a new value of \( \theta \), then the change in the log likelihood is lower bounded by the change in the \( Q \) function. This is formally stated as

\[
\log p(y|\theta) - \log p(y|\theta') \geq Q(\theta; \theta') - Q(\theta'; \theta'). \quad (12.8)
\]

The proof of this key result is then quite simple.

\[
\log p(y|\theta) - \log p(y|\theta') = Q(\theta; \theta') + H(\theta; \theta') - [Q(\theta'; \theta') + H(\theta'; \theta')] = Q(\theta; \theta') - Q(\theta'; \theta') + H(\theta; \theta') - H(\theta'; \theta') \geq Q(\theta; \theta') - Q(\theta'; \theta')
\]

In fact, by rearranging the terms of equation (12.8), we obtain the defining property of a surrogate function, as defined in equation (8.7) of Chapter 8.

\[
\log p(y|\theta) \geq Q(\theta; \theta') - Q(\theta'; \theta') + \log p(y|\theta') . \quad (12.9)
\]

So we see that the function \( Q(\theta; \theta') \) is a surrogate function for the maximization of \( \log p(y|\theta) \). As with any surrogate function, optimization of \( Q \) ensures optimization of the likelihood; so we have the following.

\[
Q(\theta; \theta') > Q(\theta'; \theta') \Rightarrow p(y|\theta) > p(y|\theta'). \quad (12.10)
\]
From this result, the central concept of the EM algorithm becomes clear. Each iteration starts with an initial parameter $\theta'$. Our objective is then to find a new parameter $\theta$ so that $Q(\theta; \theta') > Q(\theta'; \theta')$. From equation (12.10), we then know that this new parameter is guaranteed to produce a larger value of the likelihood.

With this understanding in mind, we can now state the two-step recursion that defines the EM algorithm.

E-step: \[ Q(\theta; \theta^{(k)}) = \mathbb{E}\left[ \log p(y, X|\theta) | Y = y, \theta^{(k)} \right] \] (12.11)

M-step: \[ \theta^{(k+1)} = \arg \max_{\theta \in \Omega} Q(\theta; \theta^{(k)}) \] (12.12)

Since each new value of the parameter $\theta^{(k)}$ is selected to maximize the $Q$ function, we know that this iteration will produce a monotone increasing sequence of log likelihood values, $\log p(y|\theta^{(k+1)}) \geq \log p(y|\theta^{(k)})$.

To summarize the results of this section, we list out the major properties associated with the EM algorithm below.

**Property 12.1. Decomposition of likelihood** - Let $X$ and $Y$ be random vectors with joint density function $p(x, y|\theta)$ for $\theta \in \Omega$. Then we have that

$$\log p(y|\theta) = Q(\theta; \theta') + H(\theta; \theta')$$

where $Q(\theta; \theta')$ and $H(\theta; \theta')$ are given by equations (12.5) and (12.6).

The $Q$ and $H$ functions then have the following properties.

**Property 12.2. Properties of $Q$ and $H$ functions** - Let $X$ and $Y$ be random vectors with joint density function $p(x, y|\theta)$ for $\theta \in \Omega$, and let the functions $Q$ and $H$ be given by equations (12.5) and (12.6). Then

- $Q(\theta; \theta')$ is a surrogate function for the maximization of $\log p(y|\theta)$.
- $H(\theta'; \theta') = \min_{\theta \in \Omega} H(\theta; \theta')$

### 12.4 EM for Gaussian Mixtures

Now that we have the basic tools of the EM algorithm, we can use them to estimate the parameters of the Gaussian mixture distribution introduced
in Section 12.2. In order to calculate the $Q$ function for this problem, we will need a more suitable expression for the joint distribution of $Y_n$ and $X_n$. Recalling that the Gaussian mixture is parameterized by

$$\theta = [\pi_0, \mu_0, \sigma_0^2, \cdots, \pi_{M-1}, \mu_{M-1}, \sigma_{M-1}^2],$$

then it is easily verified that the following equalities holds

$$\log p(y_n, x_n|\theta) = \log \left\{ p(y_n|x_n, \sigma_{x_n}) \pi_{x_n} \right\} = \sum_{m=0}^{M-1} \delta(x_n - m) \left\{ \log p(y_n|\mu_m, \sigma_m) + \log \pi_m \right\},$$

where $p(y_n|\mu_m, \sigma_m)$ denotes a Gaussian density with mean $\mu_m$ and standard deviation $\sigma_m$. Notice, that the delta function in the sum is only 1 when $x_n = m$, otherwise it is zero. Since the values of $Y_n$ and $X_n$ are assumed independent for different $n$, we know that

$$\log p(y, x|\theta) = \sum_{n=1}^{N} \log p(y_n, x_n|\theta) = \sum_{n=1}^{N} \sum_{m=0}^{M-1} \delta(x_n - m) \left\{ \log p(y_n|\mu_m, \sigma_m) + \log \pi_m \right\}. $$

Using this expression, we can now calculate the $Q$ function.

$$Q(\theta; \theta') = \mathbb{E}[\log p(y, X|\theta)|Y = y, \theta']$$

$$= \mathbb{E} \left[ \sum_{n=1}^{N} \sum_{m=0}^{M-1} \delta(X_n - m) \left\{ \log p(y_n|\mu_m, \sigma_m) + \log \pi_m \right\} \mid Y = y, \theta' \right]$$

$$= \sum_{n=1}^{N} \sum_{m=0}^{M-1} \mathbb{E}[\delta(X_n - m)|Y = y, \theta'] \left\{ \log p(y_n|\mu_m, \sigma_m) + \log \pi_m \right\}$$

$$= \sum_{n=1}^{N} \sum_{m=0}^{M-1} P\{X_n = m | Y = y, \theta' \} \left\{ \log p(y_n|\mu_m, \sigma_m) + \log \pi_m \right\}$$

Now plugging in the explicit expression for the Gaussian distribution, we get the final expression for the $Q$ function.

$$Q(\theta; \theta') = \sum_{n=1}^{N} \sum_{m=0}^{M-1} \left\{ \frac{-1}{2\sigma_m^2} (y_n - \mu_m)^2 - \frac{1}{2} \log (2\pi \sigma_m^2) + \log \pi_m \right\} P\{X_n = m | Y = y, \theta' \}$$
The posterior conditional probability, \( P\{X_n = m | Y = y, \theta'\} \), can be easily computed using Bayes rule. If we define the new notation that

\[
f(m|y_n, \theta') = P\{X_n = m | Y = y, \theta'\},
\]

then this function is given by

\[
f(m|y_n, \theta') = \frac{1}{\sqrt{2\pi\sigma_m'^2}} \exp \left\{ -\frac{1}{2\sigma_m'^2} (y_n - \mu_m')^2 \right\} \pi_m'.
\]

While this expression for \( f(m|y_n, \theta') \) may appear complex, it is quite simple to compute numerically.

In order to calculate the associated M-step, we must maximize \( Q(\theta; \theta') \) with respect to the parameter \( \theta \). This maximization procedure is much the same as is required for ML estimation of parameters. So for example, \( \mu_k \) and \( \sigma_k \) are calculated the same way as is used for the ML estimate of mean and variance of a Gaussian random variable. This approach results in the final EM update equations for this clustering problem.

\[
\hat{N}^{(k+1)}_m = \sum_{n=1}^{N} f(m|y_n, \theta^{(k)}) \tag{12.13}
\]

\[
\hat{\pi}^{(k+1)}_m = \frac{\hat{N}^{(k+1)}_m}{N} \tag{12.14}
\]

\[
\hat{\mu}^{(k+1)}_m = \frac{1}{\hat{N}^{(k+1)}_m} \sum_{n=1}^{N} y_n f(m|y_n, \theta^{(k)}) \tag{12.15}
\]

\[
\left[ \hat{\sigma}^2_m \right]^{(k+1)} = \frac{1}{\hat{N}^{(k+1)}_m} \sum_{n=1}^{N} \left( y_n - \hat{\mu}^{(k+1)}_m \right)^2 f(m|y_n, \theta^{(k)}) \tag{12.16}
\]

### 12.5 Algorithmic Implementation of EM Clustering

In order to better understand the EM algorithm, it is useful to take a closer look at its algorithmic implementation. To do this, we will use the algorithmic notation of pseudo code programming where \( \theta \) is the variable that contains the
current value of the parameter. Also, let $P_{n,m}$ be the program variable that contains the current value of the posterior probability $P\{X_n=m|Y=y, \theta\}$.

Using these variables, the E-step is computed by evaluating the following expression for all values of $n$ and $m$.

$$E\text{-step: } P_{n,m} \leftarrow \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp \left\{-\frac{1}{2\sigma_m^2} (y_n - \mu_m)^2 \right\} \pi_m \sum_{j=0}^{M-1} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp \left\{-\frac{1}{2\sigma_j^2} (y_n - \mu_j)^2 \right\} \pi_j$$

Notice that $\leftarrow$ indicates that the value is assigned to the program variable $P_{n,m}$. Once this is computed, we can update the model parameters by evaluating the following expressions for all values of $m$.

$$M\text{-step: } N_m \leftarrow \sum_{n=1}^{N} P_{n,m} \tag{12.17}$$

$$\pi_m \leftarrow \frac{N_m}{N} \tag{12.18}$$

$$\mu_m \leftarrow \frac{1}{N_m} \sum_{n=1}^{N} y_n P_{n,m} \tag{12.19}$$

$$\sigma_m^2 \leftarrow \frac{1}{N_m} \sum_{n=1}^{N} (y_n - \mu_m)^2 P_{n,m} \tag{12.20}$$

In this form, the meaning of the EM algorithm starts to become more clear as shown in Figure 12.3. With each update, we use the current estimate of the parameter, $\theta$, to compute the matrix $P_{n,m}$, the probability that $X_n$ has label $m$. These posterior probabilities are then used to assign each data point, $Y_n$, to the associated $M$ clusters. Since the assignment is soft, each data point has partial membership in each cluster.

Once the data points are assigned to the clusters, then the parameters of each cluster, $[\pi_m, \mu_m, \sigma_m^2]$, can be updated based on the weighted contributions of each sample. This results in an updated value for the parameter $\theta$. However, with this new parameter, we must compute new entries for the matrix $P_{n,m}$, and the iterative process repeats.

In fact, this is a generalization of the heuristic method described in the beginning of the chapter, except the hard classification heuristic of (12.1) is
replaced with the soft classification of the posterior probability. So the E-step can be viewed as a form of soft classification of the data to the available categories, while the M-step can be viewed as ML parameter estimation given those soft classifications.

12.6 EM Convergence and Majorization

In this section, we provide some insight into the convergence properties of the EM algorithm. From Property [12.2], we know that $Q(\theta; \theta')$ is a surrogate function for the maximization of the log likelihood. So we know that the iterations of the EM algorithm will produce a monotone non-decreasing sequence of log likelihoods.

However, this raises the question, does the EM algorithm actually converge to the ML estimate of $\theta$? It is difficult to give a simple and clear answer to this question because in optimization (as in life) many things can go wrong. For example, the iterations can become trapped in a local maximum of the likelihood that is not a global maximum. Even if the likelihood has no local maximum, there are strange technical problems that can occur, so it is difficult to make a simple statement about the convergence to the ML estimate.
that is absolutely true under all conditions.

But with that said, as a practical matter, the EM algorithm generally
does converge to a local maximum of the log likelihood in typical estima-
tion problems. For interested readers, the two references [72, 55] provide a
more detailed introduction to the theory of convergence for the EM algo-
rrithm. However, in order to give some intuition regarding the convergence,
we present some simplified results to illustrate the methods of proof.

Define the function $L(\theta) = \log p(y|\theta)$, then we know that for each iteration
of the EM algorithm

$$L(\theta^{(k+1)}) \geq L(\theta^{(k)}) .$$

If the ML estimate exists, then for all $k$, we also know that $L(\theta^{(k)}) \leq L(\theta_{ML}) < \infty$. Therefore, the sequence $L(\theta^{(k)})$ must be a monotone increasing sequence that is bounded above, so it must therefore reach a limit which we denote by $L^* = \lim_{k \to \infty} L(\theta^{(k)})$. However, even though the limit exists, proving that the value $L^*$ is the maximum of the likelihood estimate, or that the $\lim_{k \to \infty} \theta^{(k)}$ exists is a much more difficult matter.

The following simplified theorem gives some insight into the convergence
of the EM algorithm. The basic result is that under normal conditions, the
EM algorithm converges to a parameter value for which the gradient of the
likelihood is zero. Typically, this means that the EM algorithm converges to
a local or global maximum of the likelihood function.

**Theorem 12.6.1.** Let $\theta^{(k)} \in \Omega$ be a sequence of parameter values generated
by the EM updates. Assume that a) $\Omega$ is a open set, b) the functions $Q$ and $H$
are continuously differentiable on $\Omega^2$, c) $\lim_{k \to \infty} \theta^{(k)} = \theta^*$ exists for $\theta^* \in \Omega$. Then

$$\nabla \log p(y|\theta^*) = 0 .$$

**Proof.** First, we know that by the nature of the M-step, the updated parameter value, $\theta^{(k+1)}$, is a global, and therefore, local maximum of the $Q$ function.
This implies that

\[
0 = \nabla Q(\theta; \theta^{(k)}) \bigg|_{\theta=\theta^{(k+1)}} \\
= \lim_{k \to \infty} \nabla Q(\theta^{(k+1)}; \theta^{(k)}) \\
= \nabla Q\left( \lim_{k \to \infty} \theta^{(k+1)}, \lim_{k \to \infty} \theta^{(k)} \right) \\
= \nabla Q(\theta^*; \theta^*) \\
= \nabla \log p(y|\theta^*),
\]

where for the last equality is the result of Property 8.9.

The assumptions of this proof are a bit artificial, but the result serves to illustrate the basic concepts of convergence. In particular, assumption a) is used to ensure that the solution does not fall on the boundary of a closed set because, if this happened, the gradient would no longer need to be zero at a global maximum.

### 12.7 Simplified Methods for Deriving EM Updates

While the direct calculation of the $Q$ function can be complicated, it is clear that there is a general pattern to the result. For example, in the clustering example of Sections 12.4 and 12.5 the final EM update equations appear much like the conventional ML estimates, except that the means are weighted by the probability that a sample is from the particular class. This pattern turns out to have an underlying explanation which can be used to make the derivation of the EM updates much simpler. In fact, it turns out that for all exponential distributions, the form of the EM update is quite simple.

In the following two sections, we derive and explain an easy method for determining the EM update equations for any exponential distribution. Armed with this technique, you will be able to write down the EM algorithm for most common distributions, without any need to calculate the dreaded $Q$ function.
12.7 Simplified Methods for Deriving EM Updates

12.7.1 Exponential Distributions and Sufficient Statistics

In order to make the EM algorithm more intuitive and simpler to derive, we first must take a bit of a detour to understand two very fundamental tools of statistical estimation: sufficient statistics and exponential families. With these two very powerful tools in hand, we will be able to dramatically simplify the derivations for the EM algorithm updates; so let’s take the time to understand these important concepts.

First, we say that \( T(Y) \in \mathbb{R}^k \) is a sufficient statistic for the family of distributions, \( p(y|\theta) \) for \( \theta \in \Omega \), if the density functions can be written in the form

\[
p(y|\theta) = h(y) g(T(y), \theta),
\]

where \( g(\cdot, \cdot) \) and \( h(\cdot) \) are any two functions.

Intuitively, the sufficient statistic, \( T(Y) \), distills into a \( k \)-dimensional vector all the information required to estimate the parameter \( \theta \) from the data \( Y \). In particular, the ML estimator of \( \theta \) must be a function of the sufficient statistic \( T(Y) \). To see this, notice that

\[
\hat{\theta}_{ML} = \arg \max_{\theta \in \Omega} \log p(y|\theta)
= \arg \max_{\theta \in \Omega} \{\log h(y) + \log g(T(y), \theta)\}
= \arg \max_{\theta \in \Omega} \log g(T(y), \theta)
= f(T(y)),
\]

for some function \( f(\cdot) \).

Example 12.1. To illustrate the concept of a sufficient statistic, let us consider the simple example of a set of i.i.d. Gaussian random variables, \( \{Y_i\}_{i=1}^N \), with distribution \( N(\mu, \sigma^2) \). For this family of distributions parameterized by \( \theta = [\mu, \sigma^2]^t \), we know that the joint density can be written as

\[
p(y|\theta) = \frac{1}{(2\pi \sigma^2)^{N/2}} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^N (y_i - \mu)^2 \right\} \quad (12.22)
= \frac{1}{(2\pi \sigma^2)^{N/2}} \exp \left\{ -S \frac{1}{2\sigma^2} + b \frac{\mu}{\sigma^2} - \frac{N\mu^2}{2\sigma^2} \right\}, \quad (12.23)
\]
where $S$ and $b$ are defined by

$$b = \sum_{i=1}^{N} y_i$$  \hspace{1cm} (12.24)

$$S = \sum_{i=1}^{N} y_i^2.$$  \hspace{1cm} (12.25)

In order to see that $T(Y) = [b, S]^t$ is a sufficient statistic, we select the two functions $g(\cdot, \cdot)$ and $h(\cdot)$ from equation (12.21) to be

$$h(y) = 1$$

$$g \left( \left[ \begin{array}{c} b \\ S \end{array} \right], \left[ \begin{array}{c} \mu \\ \sigma^2 \end{array} \right] \right) = \frac{1}{(2\pi \sigma^2)^{N/2}} \exp \left\{ -S \frac{1}{2\sigma^2} + b \frac{\mu}{\sigma^2} - \frac{N\mu^2}{2\sigma^2} \right\}.$$  \hspace{1cm} (12.26)

In practice, we know that we do not need all the data in $Y$ to estimate $\mu$ and $\sigma$. For example, we can compute the ML estimator of these parameters from the sufficient statistics $b$ and $S$ as follows.

$$\hat{\mu} = \frac{b}{N}$$

$$\hat{\sigma}^2 = \frac{S}{N} - \left( \frac{b}{N} \right)^2.$$  

Many commonly used distributions, such as Gaussian, exponential, Poisson, Bernoulli, and binomial, have a structure which makes them particularly useful. These distributions are known as exponential families because their density can be written in a particularly useful exponential form. Below we give the formal definition of an exponential family of distributions.

**Definition 13. Exponential Family**

A family of density functions $p(y|\theta)$ for $y \in \mathbb{R}^N$ and $\theta \in \Omega$ is said to be a **k-parameter exponential family** if there exist functions $\eta(\theta) \in \mathbb{R}^k$, $s(y)$, $d(\theta)$ and statistic $T(y) \in \mathbb{R}^k$ such that

$$p(y|\theta) = \exp \{ \langle \eta(\theta), T(y) \rangle + d(\theta) + s(y) \}$$  \hspace{1cm} (12.26)

for all $y \in \mathbb{R}^N$ and $\theta \in \Omega$ where $\langle \cdot, \cdot \rangle$ denotes an inner product. We refer to $T(y)$ as the **natural sufficient statistic** or **natural statistic** for the exponential distribution.
Exponential families are extremely valuable because the log of their associated density functions form an inner product that is easily manipulated when computing ML parameter estimates. In general, ML estimates for exponential families have the form

\[
\hat{\theta}_{ML} = \arg \max_{\theta \in \Omega} \log p(y|\theta) = \arg \max_{\theta \in \Omega} \{ \langle \eta(\theta), T(y) \rangle + d(\theta) \} = f(T(y)),
\]

for some function \( f(\cdot) \).

Referring back to the Example 12.1, we can see that the i.i.d. Gaussian distribution of equation (12.22) is an example of an exponential family with natural sufficient statistic \( T(y) = [b, S]^t \) where \( b \) and \( S \) are defined in equations (12.24) and (12.25). More specifically, if we define

\[
\eta(\theta) = \begin{bmatrix} \frac{\mu}{\sigma^2} \\ -\frac{1}{2\sigma^2} \end{bmatrix},
\]

\[
d(\theta) = -\frac{N}{2} \left\{ \frac{\mu^2}{\sigma^2} + \log(2\pi\sigma^2) \right\},
\]

then the probability density of equation (12.22) has the form

\[
p(y|\theta) = \exp \left\{ \eta(t(\theta))T(Y) + d(\theta) \right\},
\]

which means that it is an exponential family of densities parameterized by \( \theta = [\mu, \sigma^2]^t \) with natural sufficient statistic \( T(Y) = [b, S]^t \).

Below we present some examples of other important exponential families, and for each family we derive the natural sufficient statistic, \( T(Y) \).

---

**Example 12.2.** Let \( \{Y_n\}_{n=1}^N \) be i.i.d. random vectors of dimension \( p \) with multivariate Gaussian distribution \( N(\mu, R) \) and parameter \( \theta = [\mu, \sigma R]^t \). Then \( p(y|\theta) \) is an exponential distribution with natural sufficient statistics

\[
b = \sum_{n=1}^N y_n,
\]

\[
S = \sum_{n=1}^N y_n y_n^t.
\]
To see why this is true, we can use the result of equation (2.6) to write the
density function for $Y$ as

$$p(y|\theta) = \frac{1}{(2\pi)^{Np/2}|R|^{-N/2}} \exp \left\{-\frac{1}{2} \left( \text{tr} \{SR^{-1}\} - 2b' R^{-1} \mu + N\mu' R^{-1} \mu \right) \right\}.$$ 

which has the form of (12.26) required by a natural sufficient statistic of the
distribution. From these sufficient statistics, we can also compute the ML
estimate, $\hat{\theta} = [\hat{\mu}, \hat{R}]$, as

$$\hat{\mu} = \frac{b}{N}$$
$$\hat{R} = \frac{(S/N) - \hat{\mu} \hat{\mu}'}. $$

Of course, i.i.d. Gaussian random variables are just a special case of this
example when the vector dimension is $p = 1$.

---

**Example 12.3.** Let $\{X_n\}_{n=0}^{N-1}$ be i.i.d. random variables which take on the
discrete values in the set $\{0, \cdots, M - 1\}$. The distribution of $X_n$ is param-
eterized by $\theta = [\pi_0, \cdots, \pi_{M-1}]$ where $P\{X_n = i\} = \pi_i$. Then $p(x|\theta)$ is an
exponential distribution with natural sufficient statistics

$$N_m = \sum_{n=1}^{N} \delta(X_n - m).$$

To see why this is true, we can write the density function as

$$p(x|\theta) = \prod_{m=0}^{M-1} \pi_m^{N_m}$$

because $N_m$ counts the number of times $X_n$ takes on the value $m$. This can
then be rewritten as

$$p(x|\theta) = \exp \left\{ \sum_{m=0}^{M-1} N_m \log \pi_m \right\},$$

which has the form of (12.26) required by a natural sufficient statistic of the
distribution. Again, from these sufficient statistics we can compute the ML
estimate, $\hat{\theta} = [\hat{\pi}_0, \cdots, \hat{\pi}_{M-1}]$, as

$$\hat{\pi}_m = \frac{N_m}{N}.$$
Example 12.4. Here we consider two random processes, $Y_n$ and $X_n$, with a structure similar to those used in the clustering problem of Sections 12.4 and 12.5, except we generalize the problem slightly by assuming that each $Y_n$ is a $p$-dimensional multivariate Gaussian random vector when conditioned on each $X_n$. Furthermore, assume the conditional distribution of each $Y_n$ given each $X_n$ is Gaussian with conditional mean $\mu_m$ and conditional covariance $R_m$ where $m = X_n$.

More specifically, we have that

$$p(y|x) = \prod_{n=1}^{N} \frac{1}{(2\pi)^{p/2}|R_{x_n}|^{-\frac{1}{2}}} \exp \left\{ -\frac{1}{2}(y_n - \mu_{x_n})^t R_{x_n} (y_n - \mu_{x_n}) \right\}$$

were $y$ and $x$ represent the entire sequence of $N$ random vectors and $N$ random variables.

Then in this case, $p(y,x|\theta)$ is an exponential distribution with natural sufficient statistics given by

$$N_m = \sum_{n=1}^{N} \delta(x_n - m) \quad (12.27)$$

$$b_m = \sum_{n=1}^{N} y_n \delta(x_n - m) \quad (12.28)$$

$$S_m = \sum_{n=1}^{N} y_n y_n^t \delta(x_n - m) . \quad (12.29)$$

The ML parameter estimate, $\hat{\theta}$, can then be computed from these sufficient statistics as

$$\hat{n}_m = \frac{N_m}{N} \quad (12.30)$$

$$\hat{\mu}_m = \frac{b_m}{N_m} \quad (12.31)$$

$$\hat{R}_m = \frac{S_m}{N_m} - \frac{b_m b_m^t}{N_m^2} . \quad (12.32)$$

The proof of these facts is left as an exercise. (See problem 4)
Of course the problem with the ML estimates of (12.30), (12.31), and (12.32) is that we may not know the labels $X_n$. This is the incomplete data problem that the EM algorithm addresses. In the next section, we will see how the EM algorithm can be easily applied for any such exponential distribution.

12.7.2 EM for Exponential Distributions

Now that we have the tools of exponential families firmly in hand, we can use them to simplify the derivation of the EM update equations. To do this, we will make the assumption that the joint distribution of $(X, Y)$ comes from an exponential family, where $Y$ is the observed or incomplete data, $X$ is the unobserved data, and $\theta$ is the parameter to be estimated by the EM algorithm.

More specifically, let the distribution of $(X, Y)$ be from an exponential family parameterized by $\theta$, then we know that

$$p(y, x|\theta) = \exp\{\langle g(\theta), T(y, x) \rangle + d(\theta) + s(y, x)\}$$

for some natural sufficient statistic $T(y, x)$. Assuming the ML estimate of $\theta$ exists, then it is given by

$$\theta_{ML} = \arg\max_{\theta \in \Omega} \{\langle g(\theta), T(y, x) \rangle + d(\theta)\}$$

$$= f(T(y, x)) \quad (12.33)$$

where $f(\cdot)$ is some function of $T(y, x)$.

Recalling the form of the $Q$ function, we have

$$Q(\theta; \theta') = \mathbb{E}[\log p(y, X|\theta)|Y = y, \theta']$$

where $Y$ is the observed data and $X$ is the unknown data. Using the assumed structure of the exponential distribution, we have that

$$Q(\theta; \theta') = \mathbb{E}[\log p(y, X|\theta)|Y = y, \theta']$$

$$= \mathbb{E}[\langle g(\theta), T(y, X) \rangle + d(\theta) + s(y, X)|Y = y, \theta']$$

$$= \langle g(\theta), \bar{T}(y) \rangle + d(\theta) + \text{const}$$.
where
\[ \bar{T}(y) = \mathbb{E}[T(y, X)|Y = y, \theta'] \]
is the conditional expectation of the sufficient statistic \( T(y, X) \), and “const” is a constant which does not depend on \( \theta \). Since our objective is to maximize \( Q \) with respect to \( \theta \), this constant can be dropped. A single update of the EM algorithm is then given by the recursion
\[
\theta'' = \arg \max_{\theta \in \Omega} Q(\theta; \theta') = \arg \max_{\theta \in \Omega} \left\{ \langle g(\theta), \bar{T}(y) \rangle + d(\theta) \right\} = f(\bar{T}(y)).
\] (12.34)

Intuitively, we see that the EM update of (12.34) has the same form as the computation of the ML estimate in equation (12.33), but with the expected value of the statistic, \( \bar{T} \), replacing the actual statistic, \( T \).

### 12.7.3 EM for Multivariate Gaussian Mixtures

To see how useful the result of the previous Section 12.7.2 can be, we apply it to extend the clustering example of Sections 12.4 and 12.5 to the problem of EM estimation for multivariate Gaussian mixture distributions. So in this case, each observation \( Y_n \in \mathbb{R}^p \) is a \( p \)-dimensional vector that is conditionally Gaussian given the unknown class, \( X_n \in \{0, \cdots, M-1\} \).

**Example 12.5.** Let \( Y_n \) and \( X_n \) be as in the previous Example 12.4 in this chapter. So \( \{Y_n\}_{n=1}^N \) are \( p \)-dimensional conditionally independent Gaussian random vectors given the class labels \( \{X_n\}_{n=1}^N \) with conditional mean and covariance given by \( \mu_m \) and \( R_m \) when \( X_n = m \); and \( X_n \) are assumed i.i.d. with \( P\{X_n = m\} = \pi_m \) for \( 0 \leq m < M - 1 \).

Then we know from Example 12.4 that \( p(y, x|\theta) \) is an exponential distribution with parameter
\[
\theta = (\pi_0, \mu_0, R_0, \cdots, \pi_{M-1}, \mu_{M-1}, R_{M-1})
\]
and natural sufficient statistics given by equations (12.27), (12.28), and (12.29).

In order to derive the EM update, we only need to replace the sufficient statistics in the ML estimate by their expected values. So to compute the EM
update of the parameter $\theta$, we first must compute the conditional expectation of the sufficient statistics. We can do this for the statistic $N_m$ as follows.

$$\tilde{N}_m = \mathbb{E} \left[ N_m \mid Y = y, \theta^{(k)} \right]$$

$$= \mathbb{E} \left[ \sum_{n=1}^{N} \delta(x_n - m) \mid Y = y, \theta^{(k)} \right]$$

$$= \sum_{n=1}^{N} \mathbb{E} \left[ \delta(x_n - m) \mid Y = y, \theta^{(k)} \right]$$

$$= \sum_{n=1}^{N} P \{ X_n = m \mid Y = y, \theta^{(k)} \}$$

Using a similar approach for all three sufficient statistics yields the E-step of

$$\tilde{N}_m = \sum_{n=1}^{N} P \{ X_n = m \mid Y = y, \theta^{(k)} \}$$

$$\tilde{b}_m = \sum_{n=1}^{N} y_n P \{ X_n = m \mid Y = y, \theta^{(k)} \}$$

$$\tilde{S}_m = \sum_{n=1}^{N} y_n y_t P \{ X_n = m \mid Y = y, \theta^{(k)} \}.$$ 

Then in order to calculate the M-step, we simply use these expected statistics in place of the conventional statistics in the equations for the ML estimator.

$$\pi^{(k+1)}_m = \frac{\tilde{N}_m}{N}$$

$$\mu^{(k+1)}_m = \frac{\tilde{b}_m}{\tilde{N}_m}$$

$$R^{(k+1)}_m = \frac{\tilde{S}_m}{\tilde{N}_m} - \frac{\tilde{b}_m \tilde{b}_m^t}{\tilde{N}_m^2}.$$ 

With each repetition of this process, we increase the likelihood of the observations. Assuming that the likelihood has a maximum\footnote{For the case of a Gaussian mixture with no lower bound on the eigenvalues of $R^{(k)}_m$, the likelihood is not bounded. However, in practice a local maximum of the likelihood usually provides a good estimate of the parameters.} and that one is not
trapped in a local maximum, then repeated application of the EM iterations will converge to the ML estimate of $\theta$.

12.8 ***Cluster Order Selection

Need to add a section on this. Use some of the writeup from the Cluster software manual.
12.9 Chapter Problems

1. Let $Y$ and $X$ be random objects with non-negative density function $p(y|\theta)$ for $\theta \in \Omega$. Then derive the following relationships that form the basis of the EM algorithm.

   a) Show that for all parameter values $\theta, \theta' \in \Omega$,
      \[
      \log p(y|\theta) = Q(\theta; \theta') + H(\theta; \theta')
      \]
      where the functions $Q$ and $H$ are defined as
      \[
      Q(\theta; \theta') = \mathbb{E}[\log p(y, X|\theta)|Y = y, \theta'] \quad (12.35)
      \]
      \[
      H(\theta; \theta') = -\mathbb{E}[\log p(X|y, \theta)|Y = y, \theta'] . \quad (12.36)
      \]

   b) Show that for all parameter values $\theta, \theta' \in \Omega$
      \[
      H(\theta; \theta') \geq H(\theta'; \theta')
      \]

   c) Show that for all parameter values $\theta, \theta' \in \Omega$
      \[
      \log p(y|\theta) - \log p(y|\theta') \geq Q(\theta; \theta') - Q(\theta'; \theta') .
      \]

2. Use the $Q$ function in Section 12.4 to calculate the solutions to the M-step shown in equations (12.14), (12.15), and (12.16).

3. Derive the expression for the multivariate Gaussian density of $p(y|\theta)$ given in Example 12.2 with the following form.
      \[
      p(y|\theta) = \frac{1}{(2\pi)^{Np/2}|R|^{-N/2}} \exp\left\{ -\frac{1}{2} \left( \text{tr}\{SR^{-1}\} - 2b^t R^{-1} \mu + N\mu^t R^{-1} \mu \right) \right\} .
      \]

4. Show that the result stated in Example 12.4 is correct.

5. Let $X$ and $Y$ be two random vectors of dimensions $N$ and $p$ respectively that are jointly distributed with a Gaussian mixture. More specifically, the column vector $Z = \begin{bmatrix} X \\ Y \end{bmatrix}$ has mixture density given by
      \[
      p(z) = \sum_{i=0}^{M-1} \pi_i f(z|\mu_i, B_i)
      \]

---

2 More generally, this can also be a probability mass function or a mixed probability density/mass function.
where \( f(z|\mu, B) \) is general notation for an \( N \) dimensional multivariate Gaussian density with mean \( \mu \in \mathbb{R}^N \) and inverse covariance \( B \in \mathbb{R}^{N \times N} \) given by

\[
f(z|\mu, B) = \frac{1}{(2\pi)^{N/2}|B|^{1/2}} \exp \left\{ -\frac{1}{2}(z - \mu)^tB(z - \mu) \right\}.
\]

Furthermore, assume that \( B_i \) has a block structure with the form

\[
B_i = \begin{bmatrix}
B_{xx,i} & B_{xy,i} \\
B_{xy,i}^t & B_{yy,i}
\end{bmatrix},
\]

where \( B_{xx,i} \) is \( N \times N \), \( B_{xy,i} \) is \( N \times p \), and \( B_{yy,i} \) is \( p \times p \).

a) Show that the distribution of \( X \) is a Gaussian mixture.

b) Show that the conditional distribution of \( X \) given \( Y \) is a Gaussian mixture with the form

\[
p(x|y) = \sum_{i=0}^{M-1} \pi_i(y) f(x|\mu_i(y), B_{xx,i}),
\]

and find expressions for its parameters \( \mu_i(y) \) and \( \pi_i(y) \).

c) Use the result of b) above to find an expression for the function \( g(y) \) so that

\[
\mathbb{E}[X|Y] = g(Y).
\]

d) Is \( g(Y) \) a linear function of \( Y \)? Why or why not?

6. Let \( \{X_n\}_{n=1}^N \) be i.i.d. random variables with \( P\{X_n = i\} = \pi_i \) for \( i = 0, \cdots, M - 1 \). Also, assume that \( Y_n \in \mathbb{R}^p \) are conditionally independent Gaussian random vectors given \( X_n \) and that the conditional distribution of \( Y_n \) given \( X_n \) is distributed as \( N(\mu_{x_n}, \gamma_{x_n}) \).

a) Derive an expression for the maximum likelihood estimates of the parameters \( \{\pi_i, \mu_i, \gamma_i\}_{i=0}^{M-1} \) given the complete data \( \{X_n, Y_n\}_{n=1}^N \).

b) Derive an expression for the posterior distribution of \( X_n \) given \( \{Y_n\}_{n=1}^N \).

c) Derive an expression for the expectation and maximization steps of the EM algorithm for estimating the parameters \( \{\pi_i, \mu_i, \gamma_i\}_{i=0}^{M-1} \) from the observations \( \{Y_n\}_{n=1}^N \).
7. Let $X_n, W_n,$ and $Y_n$ each be i.i.d. discrete time random processes with

$$Y_n = X_n + W_n$$

where $X_n \sim N(\mu, R)$ and $W_n \sim N(0, I)$.

a) Show that given $N$ realizations, $X_1, X_2, \cdots, X_N$, the ML estimates of $R$ and $\mu$ are

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} X_n$$

and

$$\hat{R} = \frac{1}{N} \sum_{n=1}^{N} (X_n - \hat{\mu})(X_n - \hat{\mu})^t.$$ 

b) Derive the EM algorithm for computing the ML estimates of $R$ and $\mu$ from $\{Y_n\}_{n=1}^{N}$.

8. Let $X_n$ be a sequence of i.i.d. random variables for $n = 1, \cdots, N$, and let $Y_n$ be a sequence of random variables that are conditionally independent and exponentially distributed with mean $\mu_i$ for $i = 0, \cdots, M - 1$ given the corresponding values of $X_n$. More specifically, the distributions of $X$ and $Y$ are given by

$$P\{X_n = i\} = \pi_i$$

$$p(y_n|x_n) = \frac{1}{\mu_n} e^{-y_n/\mu_n} u(y_n)$$

where $i \in \{0, \cdots, M - 1\}$ and $u(\cdot)$ is the unit step function. Furthermore, let $\theta = [\pi_0, \mu_0, \cdots, \pi_{M-1}, \mu_{M-1}]$ be the parameter vector of the distribution.

a) Derive a closed form expression for the ML estimate of $\theta$ given both $X$ and $Y$ (i.e., the complete data).

b) Find the natural sufficient statistics for the exponential distribution of $(X, Y)$.

c) Use the results of this chapter to derive the EM algorithm for computing the ML estimate of $\theta$ from $Y$. 

9. Let $\{X_n\}_{n=1}^N$ be i.i.d. random variables with distribution

$$P\{X_n = m\} = \pi_m,$$

where $\sum_{m=0}^{M-1} \pi_m = 1$. Also, let $Y_n$ be conditionally independent random variables given $X_n$, with Poisson conditional distribution

$$p(y_n|x_n = m) = \frac{\lambda^y_m e^{-\lambda_m}}{y_n!}.$$

a) Write out the density function for the vector $Y$.

b) What are the natural sufficient statistics for the complete data $(X, Y)$?

c) Give an expression for the ML estimate of the parameter

$$\theta = (\pi_0, \lambda_0, \cdots, \pi_{M-1}, \lambda_{M-1}),$$
given the complete data $(X, Y)$.

d) Give the EM update equations for computing the ML estimate of the parameter $\theta = (\pi_0, \lambda_0, \cdots, \pi_{M-1}, \lambda_{M-1})$ given the incomplete data $Y$. 
The Expectation-Maximization (EM) Algorithm
Chapter 13

Markov Chains and Hidden Markov Models

In this chapter, we will introduce the concept of Markov chains and show how Markov chains can be used to model signals using structures such as hidden Markov models (HMM). Markov chains represent data as a sequence of states in time, and they are based on the simple idea that each new state is only dependent on the previous state. This simple assumption makes them easy to analyze, but still allows them to be very powerful tools for modeling a wide variety of physical processes.

We finish the chapter by discussing what happens to the states of a homogeneous Markov chain as time tends towards infinity. Once the transient behavior decays, one would expect that a homogeneous Markov chain should reach steady state. When this does happen, we call the Markov chain ergodic, and we will present some simple technical conditions that ensure ergodicity. In fact, we will see that ergodic Markov chains are very useful in a wide range of applications and are particularly important in applications such as Monte Carlo simulation, generating a stochastic sample of a Markov random field, and stochastic optimization. This issue of stochastic sampling and optimization will be considered in more detail in Chapter 15.

13.1 Markov Chains

It is often useful to model the discrete state of a real-world system as it evolves in time. So for example, the discrete-time random process, \( \{X_n\}_{n=0}^N \), which
takes values on a countable set, Ω, might be used to represent many things. For a speech signal, each state could represent the particular phoneme being voice; for music, it could represent the specific note of a discrete musical scale; and for a digitally modulated signal, it could represent the discrete binary values 0 or 1.

While the behavior of physical systems can never be fully characterized by a stochastic model, it is often useful to approximate the distribution of each new state, $X_n$, as only dependent on the previous state, $X_{n-1}$. In this case, we call the random process a Markov chain. The following definition makes this more precise.

**Definition 14. Markov Chain**

Let $\{X_n\}_{n=0}^\infty$ be a discrete-time random process taking values in the countable set, Ω. Then we say that $X_n$ is a **Markov chain** if for all $j \in \Omega$ and for all $n > 0$

$$P\{X_n = j|X_k \text{ for all } k < n\} = P\{X_n = j|X_{n-1}\},$$

where Ω is the state space of the Markov chain.

Since the state space is discrete and countable, we can assume, without loss of generality, that $\Omega = \{0, \cdots, M-1\}$, where $M = \infty$ is allowed. We also allow for the possibility that the Markov chain is of finite length, so that the Markov chain consists of the sequence $X_0, \cdots, X_N$. In the case where $\Omega$ is continuously valued, then we refer to $X_n$ as a **discrete-time Markov process**. However, we will primarily focus on Markov chains because they are easy to analyze and of great practical value. Nonetheless, most of the results we derive are also true for discrete-time Markov processes.

In order to analyze a Markov chain, we need to first define some basic notation. The marginal distribution, $\pi_j^{(n)}$, and transition probabilities, $P_{i,j}^{(n)}$, of the Markov chain are defined as

$$\pi_j^{(n)} \triangleq P\{X_n = j\}$$
$$P_{i,j}^{(n)} \triangleq P\{X_n = j|X_{n-1} = i\}$$

for $i, j \in \Omega$. If the transition probabilities, $P_{i,j}^{(n)}$, do not depend on time, $n$, then we say that $X_n$ is a **homogeneous Markov chain**. For the remainder
of this chapter, we will assume that Markov chains are homogeneous unless otherwise stated.

One might initially guess that a homogeneous Markov chain will have a stationary distribution; so that for example, its marginal distribution will not vary with time. However, this is not true, since the Markov chain can have transient behavior depending on the initial distribution of states at \( n = 0 \). Nonetheless, it is reasonable to expect that after a sufficient time, any transient behavior would die out, and the Markov chain might reach some type of steady-state behavior. In fact, Section 13.4 will treat this issue in detail.

From the parameters of the Markov process, we can derive an expression for the probability of the sequence \( \{X_n\}_{n=0}^N \). In order to simplify notation, denote the distribution of the initial state as

\[
\tau_j = \pi_j^{(0)} = P\{X_0 = j\}.
\]

Then the probability of a particular state sequence for the Markov chain is given by the product of the probability of the initial state, \( \tau_{x_0} \), with the probabilities for each of the \( N \) transitions from state \( x_{n-1} \) to state \( x_n \). This product is given by

\[
p(x) = \tau_{x_0} \prod_{n=1}^N P_{x_{n-1},x_n}.
\]

(13.1)

From this expression, it is clear that the Markov chain is parameterized by its initial distribution, \( \tau_j \), and its transition probabilities, \( P_{i,j} \).

---

**Example 13.1.** Let \( \{X_n\}_{n=0}^N \) be a Markov chain with state-space \( \Omega = \{0, 1\} \) and parameters given by

\[
\begin{align*}
\tau_j &= \frac{1}{2} \\
P_{i,j} &= \begin{cases} 
1 - \rho & \text{if } j = i \\
\rho & \text{if } j \neq i
\end{cases}
\end{align*}
\]

This Markov chain starts with an equal chance of being 0 or 1. Then with each new state, it has probability \( \rho \) of changing states, and probability \( 1 - \rho \) of remaining in the same state. If \( \rho \) is small, then the probability of changing state is small, and the Markov chain is likely to stay in the same state for
an extended period of time. When $\rho = 1/2$ each new state is independent of the previous state; and when $\rho$ is approximately 1, then the state is likely to change with each new value of $n$. These three cases are illustrated in Figure [13.1]

If we define the statistic

$$K = N - \sum_{n=1}^{N} \delta(X_n - X_{n-1}) ,$$

then $K$ is the number of times that the Markov chain changes state. Using this definition, we can express the probability of the sequence as

$$p(x) = (1/2)(1 - \rho)^{N-K}\rho^K .$$

### 13.2 Parameter Estimation for Markov Chains

Markov chains are very useful for modeling physical phenomena because, as with AR models, we simply predict the distribution of the next sample from the previous one. However, the challenge with Markov chains is often to accurately estimate the parameters of the model from real data. If the Markov chain is nonhomogeneous, then the number of parameters will grow with the length of the sequence. So it is often practically necessary to assume
Markov chains are homogeneous. However, even when the Markov chain is homogeneous, there are $M^2$ parameters to choose, where $M = |\Omega|$ is the number of states.\footnote{The quantity $M^2$ results from the sum of $M$ parameters for the initial state plus $M(M - 1)$ parameters for the transition probabilities. The value $M(M - 1)$ results form the fact that there are $M$ rows to the transition matrix and each row has $M - 1$ degrees of freedom since it must sum to 1.} Therefore, it is important to have effective methods to estimate these parameters.

Fortunately, we will see that Markov chains are exponential distributions, so parameters are easily estimated from natural sufficient statistics (See Section 12.7.1 for details). Let $\{X_n\}_{n=0}^N$ be a Markov chain parameterized by $\theta = [\tau_j, P_{i,j}: \text{for } i, j \in \Omega]$, and define the statistics

$$N_j = \delta(X_0 - j)$$
$$K_{i,j} = \sum_{n=1}^N \delta(X_n - j)\delta(X_{n-1} - i).$$

The statistic $K_{i,j}$ essentially counts the number of times that the Markov chain transitions from state $i$ to $j$, and the statistic $N_j$ counts the number of times the initial state has value $j$. Using these statistics, we can express the probability of the Markov chain sequence as

$$p(x) = \left( \prod_{j \in \Omega} \tau_j^{N_j} \right) \left( \prod_{i \in \Omega} \prod_{j \in \Omega} P_{i,j}^{K_{i,j}} \right),$$

which means that the log likelihood has the form\footnote{This expression assumes that the parameters are non-zero, but the results for the ML parameter estimates still hold in the general case.}

$$\log p(x) = \sum_{j \in \Omega} \left\{ N_j \log(\tau_j) + \sum_{i \in \Omega} K_{i,j} \log(P_{i,j}) \right\}. \quad (13.2)$$

Based on this, it is easy to see that the distribution of the Markov chain $X_n$ with parameter $\theta$ is from an exponential family, and that $N_i$ and $K_{i,j}$ are its natural sufficient statistics. From (13.2), we can derive the maximum likelihood estimates of the parameter $\theta$ in much the same manner as is done for the ML estimates of the parameters of a Bernoulli sequence. This results
Figure 13.2: This diagram illustrates the dependency of quantities in a hidden Markov model. Notice that the values $X_n$ form a Markov chain in time, while the observed values, $Y_n$, are dependent on the corresponding labels $X_n$.

in the ML estimates

\[ \hat{\gamma}_j = \frac{N_j}{\sum_{j \in \Omega} K_{i,j}} \]  \hspace{1cm} (13.3)

\[ \hat{P}_{i,j} = \frac{K_{i,j}}{\sum_{j \in \Omega} K_{i,j}} \] \hspace{1cm} (13.4)

So the ML estimate of transition parameters is quite reasonable. It simply counts the rate at which a particular transition occurs from state $i$ to state $j$, and then normalizes this quantity so it sums to 1 for each initial state $i$.

### 13.3 Hidden Markov Models

One important use of Markov chains is in hidden Markov models (HMM). Figure 13.2 shows the structure of an HMM. The discrete values $\{X_n\}_{n=0}^N$ form a Markov chain in time, and their values determine the distribution of the corresponding observations $\{Y_n\}_{n=1}^N$. Much like in the case of the Gaussian mixture distribution, the labels $X_n$ are typically not observed in the real application, but we can imagine that their existence explains the changes in behavior of $Y_n$ over long time scales. As with the example of the mixture distribution from Section 12.2, the HMM model is a doubly-stochastic model because the unobserved stochastic process $X_n$ controls the observed stochastic process $Y_n$.

HMMs are very useful in applications such as audio or speech processing. For these applications, the observations $Y_n$ are typically continuously valued feature vectors describing the local sound properties, and the discrete random variables, $X_n$, represent the underlying state of the audio signal. So for example in speech applications, $X_n$ might be a label that represents the
specific phoneme or sound that is being voiced in the pronunciation of a word.

In order to analyze the HMM, we start by defining some notation. Let the density function for $Y_n$ given $X_n$ be given by

$$f(y|k) = P\{Y_n \in dy|X_n = k\},$$

and let the Markov chain $X_n$ be parameterized by $\theta = [\tau_j, P_{i,j}: \text{for } i, j \in \Omega]$, then the joint density function for the sequences $Y$ and $X$ is given by

$$p(y, x|\theta) = \tau_{x_0} \prod_{n=1}^{N} \{f(y_n|x_n)P_{x_{n-1},x_n}\}.$$

and assuming that no quantities are zero, the log likelihood is given by

$$\log p(y, x|\theta) = \log \tau_{x_0} + \sum_{n=1}^{N} \{\log f(y_n|x_n) + \log P_{x_{n-1},x_n}\}.$$

There are two basic tasks which one typically needs to perform using HMMs. The first task is to estimate the unknown states, $X_n$, from the observed data, $Y_n$, and the second task is to estimate the parameters, $\theta$, from the observations, $Y_n$. The following two sections explain how these problems can be solved using dynamic programing and the EM algorithm of Chapter 12.

### 13.3.1 State Sequence Estimation and Dynamic Programming

When using HMMs, it is often important to be able to estimate the unobserved discrete state sequence, $X$, from the full set of observations, $Y$. This problem is known as state sequence estimation. While different estimators present different strengths and weaknesses, one commonly used estimate for the state sequence is the MAP estimate given by

$$\hat{x} = \arg \max_{x \in \Omega^N} p(x|y, \theta) = \arg \max_{x \in \Omega^N} \log p(y, x|\theta).$$

---

3This is actually a mixed probability density and probability mass function. This is fine as long as one remembers to integrate over $y$ and sum over $x$. 
When we expand out the MAP optimization problem using the log likelihood of the HMM, it is not obvious that it can be tractably solved.

\[
\hat{x} = \arg \max_{x \in \Omega^N} \left\{ \log \tau_{x_0} + \sum_{n=1}^{N} \{ \log f(y_n|x_n) + \log P_{x_{n-1},x_n} \} \right\} \tag{13.5}
\]

On its face, the optimization of (13.5) is an extraordinarily difficult problem, requiring a search over a state space of the size \(M^{N+1}\). However, the problem’s complexity can be dramatically reduced by decomposing it into components corresponding to the past, present, and future values of the states and observations. Figure 13.3 provides a visual representation of how these observations and states are related along with a table defining some new notation for the problem. This notation will allow us to more compactly express the probabilities of past, present, and future events.

Using this new notation and the assumed dependencies of the HMM, we can write the joint distribution of the observations and states as the following product, (See problem 2)

\[
p(y, x) = p(y_{>n}, x_{>n}|x_n)f(y_n|x_n)p(y_{<n}, x_n, x_{<n}) , \tag{13.6}
\]

where the conditional distributions have the following intuitive interpreta-
13.3 Hidden Markov Models

Using this formulation of the problem, we may define a solution to the future portion of the problem given an assumed value for \( x_n = j \) as the function

\[
L(j, n) = \max_{x_n > n} \{ \log p(y_n | x_n = j) \}.
\]

(13.7)

So the function \( L(x_n, n) \) specifies the maximum log probability of the future states and observations given that the present state at time \( n \) is known to be \( x_n \). From this definition, we can work out the following recursive relationship for \( L(i, n - 1) \) in terms of a maximization involving \( L(j, n) \).

\[
L(i, n - 1) = \max_{j \in \Omega} \{ \log f(y_n | j) + \log P_{i,j} + L(j, n) \}
\]

(13.7)

The application of this recursion, along with the additional boundary constraint that \( L(j, N) = 0 \), is a special case of a more general optimization strategy known as **dynamic programming**. Using this approach, the values of \( L(i, n) \) can be computed for \( 0 \leq n \leq N \) and \( i \in \Omega \) with order \( NM^2 \) computational complexity. The complexity result is due to the fact that for each value of \( n \) and for each value of \( i \), we must maximize over \( M \) possible values of \( j \).

Of course, the values of \( L(i, n) \) are not actually what we are needed. What we need to know are the values of the states, \( x_n \), that maximize the likelihood. However, with the stored values of \( L(i, n) \) from the dynamic programing recursion, we can easily compute the states using the following relationships.

\[
\hat{x}_0 = \arg \max_{j \in \Omega} \{ \log \tau_j + L(j, 0) \}
\]

(13.8)

\[
\hat{x}_n = \arg \max_{j \in \Omega} \{ \log P_{\hat{x}_{n-1},j} + \log f(y_n | j) + L(j, n) \}
\]

(13.9)

Figure[13.4] shows the complete algorithm for MAP sequence estimation.

### 13.3.2 State Probability and the Forward-Backward Algorithm

While the MAP estimate of the state sequence, \( x_n \), is useful, in some cases we may need to know the actual distribution of individual states. In particular,
MAP Sequence Estimation:

For \( i \in \Omega \), \( L(i, N) = 0 \)

For \( n = N \) to 1 {
    For \( i \in \Omega \),
    \[
    L(i, n-1) = \max_{j \in \Omega} \{ \log f(y_n | j) + \log P_{i,j} + L(j, n) \}
    \]
}

For \( i \in \Omega \), \( \hat{x}_0 = \arg \max_{j \in \Omega} \{ \log \tau_k + L(j, 0) \} \)

For \( n = 1 \) to \( N \) {
    For \( i \in \Omega \),
    \[
    \hat{x}_n = \arg \max_{j \in \Omega} \{ \log P_{\hat{x}_{n-1}, j} + \log f(y_n | j) + L(j, n) \}
    \]
}

Figure 13.4: A pseudocode specification of the dynamic programming algorithm used for MAP state sequence estimation of HMMs.

we will need to know the joint distribution of pairs of states, \((x_{n-1}, x_n)\), in order to train HMM models later is Section [13.3.3](#).

In fact, we can use techniques very similar to those of Section [13.3.1](#) in order to compute the marginal and joint distributions of states. Again, the trick is to break up the observation and state variables into their past, present, and future components. To do this, we define the following two functions.

\[
\begin{align*}
\alpha_n(j) &= p(x_n = j, y_n, y_{<n}) \\
\beta_n(j) &= p(y_{>n} | x_n = j)
\end{align*}
\]

If we can compute these two functions, then the joint probability of the observations and \( x_n \) can be expressed simply as

\[
p(x_n = j, y) = \alpha_n(j)\beta_n(j), \quad (13.10)
\]

and from this it is easy to compute the conditional density of \( x_n \) as,

\[
p(x_n = j | y) = \frac{\alpha_n(j)\beta_n(j)}{p(y)}, \quad (13.11)
\]

where \( p(y) \) is just a normalizing constant given by

\[
p(y) = \sum_{j \in \Omega} \alpha_0(j)\beta_0(j) = \sum_{j \in \Omega} \alpha_0(j)\beta_0(j). \quad (13.12)
\]
We can also compute the joint conditional density of \((x_{n-1}, x_n)\) similarly as

\[
p(x_{n-1} = i, x_n = j | y) = \frac{\alpha_{n-1}(i) P_{i,j} f(y_n | j) \beta_n(j)}{p(y)},
\]

(13.13)

where again \(p(y)\) can be computed using equation (13.12).

The efficient algorithm for computing the functions \(\alpha_n(j)\) and \(\beta_n(j)\) is known as the forward-backward algorithm. It takes its name from the fact that \(\alpha_n(j)\) is computed using a forward recursion, and \(\beta_n(j)\) is computed using a backward recursion. The overall flow of the forward-backward algorithm is very similar to that of the dynamic programming technique used in Section 13.3.1. The forward recursion is given by

\[
\alpha_n(j) = \sum_{i \in \Omega} \alpha_{n-1}(i) P_{i,j} f(y_n | j),
\]

(13.14)

with a boundary condition of \(\alpha_0(j) = \tau_j\), and the backward recursion is given by

\[
\beta_n(i) = \sum_{j \in \Omega} P_{i,j} f(y_{n+1} | j) \beta_{n+1}(j)
\]

(13.15)

with a boundary condition of \(\beta_N(i) = 1\). Figure 13.5 gives a pseudocode summary of the forward backward algorithm. Again, the total computation required for this algorithm is order \(NM^2\) due to the requirement of computing the functions for all \(n\) and \(i\), and the sum of the index \(j\).

### 13.3.3 Training HMMs with the EM Algorithm

Of course, HMMs are only useful if they accurately model the behavior of real physical processes. Since the HMM represents a generic model structure, it is necessary to fit the properties of a specific HMM model to some specific physical process of interest. In practice, this fitting process is typically accomplished by estimating the parameters of the HMM from some typical examples. Often, these examples sequences, known as training data, are collected for the specific purpose of training the HMM model.

Ideally, the training data should consist of sequences of observations, \(Y_n\), along with the hidden states, \(X_n\). This might be possible in cases where the exact states, \(X_n\), can be measured at additional cost. However in many cases,
Forward-Backward Algorithm:

/* forward step */
For $j \in \Omega$, $\alpha_0(j) = \tau_j$
For $n = 1$ to $N$
    For $j \in \Omega$,
    $$\alpha_n(j) = \sum_{i \in \Omega} \alpha_{n-1}(i)P_{i,j}f(y_n|j)$$

/* backwards step */
For $i \in \Omega$, $\beta_N(i) = 1$
For $n = N$ to $1$
    For $i \in \Omega$,
    $$\beta_n(i) = \sum_{j \in \Omega} P_{i,j}f(y_{n+1}|j)\beta_{n+1}(j)$$

/* calculate probabilities */
For $n = 1$ to $N$ and $i, j \in \Omega$,
    $$p(y) = \sum_{i \in \Omega} \alpha_0(i)\beta_0(i)$$
    $$P\{X_n = i|Y = y, \theta\} = \frac{\alpha_n(i)\beta_n(i)}{p(y)}$$
    $$P\{X_n = j, X_{n-1} = i|Y = y, \theta\} = \frac{\alpha_{n-1}(i)P_{i,j}f(y_n|j)\beta_n(i)}{p(y)}$$

Figure 13.5: A pseudocode specification of the forward-backward algorithm required for computing the marginal and conditional distributions of states in an HMM.

the unknown states, $X_n$, may not be available at any cost. For example, in speech modeling, $X_n$ could represent the phoneme being voiced and $Y_n$ the spectral features of the auditory signal. In this example, there is no way to directly measure the phoneme being voiced, so any training must estimate the parameters of the HMM model while simultaneously estimating the unknown states, $X_n$. This problem of parameter estimation from incomplete data is a classic example for which EM the algorithm, as described Chapter 12, is perfectly suited.

In a typical application, the observed quantity $Y_n$ is an $p$ dimensional multivariate Gaussian random vector with conditional distribution $N(\mu_{x_n}, R_{x_n})$. So the complete set of parameters for the HMM will be $\theta = [\mu_j, R_j, \tau_j, P_{i,j} : \text{for } i, j \in \Omega]$. In this case it can be shown that the joint distribution, $p(x, y|\theta)$,
is an exponential distribution with natural sufficient statistics given by

\[ b_j = \sum_{n=1}^{N} y_n \delta(x_n - j) \] (13.16)

\[ S_j = \sum_{n=1}^{N} y_n y_n^t \delta(x_n - j) \] (13.17)

\[ N_j = \delta(x_0 - j) \] (13.18)

\[ K_{i,j} = \sum_{n=1}^{N} \delta(x_n - j) \delta(x_{n-1} - i) \] , (13.19)

and another useful statistic, \( K_j \), can be computed from these as

\[ K_j = \sum_{i \in \Omega} K_{i,j} = \sum_{n=1}^{N} \delta(x_n - j) . \]

Referring back to the ML estimates for multivariate Gaussian distributions given in equations (2.8) and (2.9), and the ML estimate derived for Markov chains in equations (13.3) and (13.4), we can write down the ML estimate of the entire parameter vector, \( \theta \), given both \( X \) and \( Y \). These are given by

\[ \hat{\mu}_j = \frac{b_j}{K_j} \]

\[ \hat{R}_j = \frac{S_j}{K_j} - \frac{b_j b_j^t}{K_j^2} \]

\[ \hat{\tau}_j = N_j \]

\[ \hat{P}_{i,j} = \frac{K_{i,j}}{\sum_{j \in \Omega} K_{i,j}} . \]

Remember, these are the closed form ML estimates that we compute when we can observe the states \( X_n \) directly. So in this case, the problem is not incomplete.

Now the results from Section 12.7 dramatically simplify our computation because they state that the EM update equations directly result from replacing the natural sufficient statistics of the complete data problem with their expected values. So to compute the EM update of the parameter \( \theta \), we first compute the conditional expectation of the sufficient statistics in the E-step.


as

\[
\bar{b}_j \leftarrow \sum_{n=1}^{N} y_n P\{X_n = j|Y = y, \theta\}
\]

\[
\tilde{S}_j \leftarrow \sum_{n=1}^{N} y_n y_n^t P\{X_n = j|Y = y, \theta\}
\]

\[
\bar{N}_j \leftarrow P\{X_0 = j|Y = y, \theta\}
\]

\[
\bar{K}_{i,j} \leftarrow \sum_{n=1}^{N} P\{X_n = j, X_{n-1} = i|Y = y, \theta\}
\]

\[
\bar{K}_j \leftarrow \sum_{n=1}^{N} P\{X_n = j|Y = y, \theta\}.
\]

Then the HMM model parameters, \( \theta = [\mu_j, R_j, \tau_j, P_{i,j} : \text{for } i, j \in \Omega] \), are updated using the M-step

\[
\hat{\mu}_j \leftarrow \frac{\bar{b}_j}{\bar{K}_j}
\]

\[
\hat{R}_j \leftarrow \frac{\tilde{S}_j}{\bar{K}_j} - \frac{\bar{b}_j \bar{b}_j^t}{\bar{K}_j^2}
\]

\[
\hat{\tau}_j \leftarrow \bar{N}_j
\]

\[
\hat{P}_{i,j} \leftarrow \frac{\bar{K}_{i,j}}{\sum_{j \in \Omega} \bar{K}_{i,j}}.
\]

Notice that the E-step requires the computation of the posterior probability of \( X_n \) given \( Y \). Fortunately, this may be easily computed using the forward-backward algorithm shown in Figure 13.5 of the previous section.

### 13.4 Stationary Distributions of Markov Chains

In addition to using Markov chains as models of data, they can also serve as a very powerful computational tool. For example, in Chapter 14 we will use Monte Carlo Markov chains techniques to generate samples from very general posterior distributions. Similar methods can also be used to numerically compute expectations/integrals or to solve optimization problems.
In order to provide a framework for these methods, we will first need to introduce a theory of ergodic Markov chains [56]. The key concept in this theory is that a well behaved Markov chain should eventually reach a stable stationary distribution after the transient behavior dies away. So we will need tools for both guaranteeing that such a steady state distribution is reached, and determining precisely what that steady state distribution is.

Again, let \( \{X_n\}_{n=0}^{\infty} \) be a homogeneous Markov chain with marginal density given by \( \pi_j(n) = P\{X_n = j\} \) and transitions probabilities given by \( P_{i,j} = P\{X_n = j|X_{n-1} = i\} \). Then a fundamental property of Markov chains is that the marginal probability density for time \( n + 1 \) can be expressed as

\[
\pi_j^{(n+1)} = \sum_{i \in \Omega} \pi_i^{(n)} P_{i,j} .
\]  

(13.20)

When the number of states is finite, then we can more compactly express these ideas using vector/matrix notation. To do this, let \( \pi(n) \) be a \( 1 \times M \) row vector containing the marginal densities, and let \( P \) be an \( M \times M \) matrix containing the transition probabilities. Then in matrix notation, we have that

\[
\pi^{(n+1)} = \pi^{(n)} P .
\]  

(13.21)

Repeated application of (13.21) results in the equation

\[
\pi^{(n+m)} = \pi^{(n)} P^m ,
\]  

(13.22)

where \( P^m \) denotes the matrix raised to the \( m^{th} \) power. So we can compute the marginal distribution at any time in the future by repeated multiplication by the transition matrix \( P \). In particular, we can compute the marginal density at any time \( n \) as

\[
\pi^{(n)} = \pi^{(0)} P^n .
\]

Intuitively, we might expect that as \( n \) grows large, the Markov chain should reach some type of steady state. If that were to hold, then we must have that \( \lim_{n \to \infty} P^n \) exists and that

\[
\pi^\infty = \lim_{n \to \infty} \pi^{(0)} P^n = \pi^{(0)} P^\infty .
\]

Moreover, we might expect that the steady state behavior should not depend on the initial distribution, \( \pi^{(0)} \). So if this holds, then we know two important
things. First, each row of $P^\infty$ must equal $\pi^\infty$, so we have that
\[ P^\infty = 1 \pi^\infty, \]
where $1$ is an $M \times 1$ vector of 1’s, and that $\pi^\infty$ must solve the equation
\[ \pi^\infty = \pi^\infty P. \quad (13.23) \]
In fact, equation (13.23) also indicates that the matrix $P$ has an eigenvalue of 1, with an associated eigenvector of $\pi^\infty$.

Unfortunately, it is not guaranteed that a homogeneous Markov chain will reach a steady state distribution that is invariant to the initial distribution. As a matter of fact, it is relatively easy to produce counterexamples, which we will do momentarily. Fortunately, the things that can go wrong can be grouped into three basic categories, and we can provide technical conditions to guard against these cases.

The three basic counter examples to steady state behavior are:

- The density can become trapped in disconnected states;
- The density can become periodic;
- The density can disperse.

Below we present three corresponding counter examples illustrating each of these behaviors.

---

**Example 13.2.** In this example, we illustrate how a homogeneous Markov chain can fail to achieve a steady state distribution.

For the first case, let the Markov chain take values in the set $\{-1, 1\}$. Let $X_{n+1} = X_n$ and let the initial distribution be given by $\pi^{(0)}$. In this case, it is obvious that the steady state distribution will always be given by
\[ \pi_i^\infty = \lim_{n \to \infty} P\{X_n = i\} = \pi_i^{(0)}. \]
Therefore, the steady state distribution is not independent of the initial distribution.
For the second case, let the Markov chain take values in the set \{-1, 1\}. Let \(X_0 = 1\), and let \(X_{n+1} = -X_n\). In this case, it can be easily shown that

\[
P\{X_n = 1\} = \begin{cases} 
1 & \text{if } n \text{ is even} \\
0 & \text{if } n \text{ is odd}
\end{cases}.
\]

Therefore, the \(\lim_{n \to \infty} P\{X_n = 1\}\) does not exist.

For the third case, let the Markov chain take on integer values. Let \(X_0 = 0\), and let \(X_{n+1} = X_n + B_n\) where \(B_n\) is an i.i.d. sequence of random variables with \(P\{B_n = 1\} = P\{B_n = -1\} = 1/2\). In this case, it can be shown that for all integers \(i\),

\[
\pi_i^\infty = \lim_{n \to \infty} P\{X_n = i\} = 0.
\]

Therefore, the density function disperses to zero.

So if such simple examples of homogeneous Markov chains can result in counter examples, how can we ensure convergence to a steady state? Of course, this is a huge field of research and many results exist, but we will provide some simple conditions, that when satisfied, ensure ergodic behavior of the Markov chain. We start with two definitions, which we will use to guard against disconnected states.

**Definition 15. Communicating states**

The states \(i, j \in \Omega\) of a discrete time discrete state homogeneous Markov chain are said to communicate if there exist integers \(n > 0\) and \(m > 0\) such that \(P_{i,j}^n > 0\) and \(P_{j,i}^m > 0\).

Intuitively, two states communicate if it is possible to transition between the two states. It is easily shown that communication is an equivalence property, so it partitions the set of states into disjoint subsets, and only states within a subset communicate with each other. This leads to a natural definition for irreducible Markov chains.

**Definition 16. Irreducible Markov Chain**

A discrete time discrete state homogeneous Markov chain is said to be irreducible if for all \(i, j \in \Omega\), \(i\) and \(j\) communicate.

So a Markov chain is irreducible if it is possible to change from any initial state to any other state in finite time.
The next condition is used to guard against periodic repetition of states that can last indefinitely and will not allow convergence to steady state.

**Definition 17. Periodic state**

We denote the period of a state \( i \in \Omega \) by the value \( d(i) \) where \( d(i) \) is the largest integer so that \( [P^n]_{i,i} = 0 \) whenever \( n \) is not divisible by \( d(i) \). If \( d(i) > 1 \), then we say that the state \( i \) is periodic.

It can be shown that states of a Markov chain that communicate must have the same period. Therefore, all the states of an irreducible Markov chain must have the same period. We say that an irreducible Markov chain is aperiodic if all the states have period 1.

Using these definitions, we may now state a theorem which gives basic conditions for convergence of the distribution of the Markov chain.

**Theorem 13.4.1. Limit Theorem for Markov Chains**

Let \( X_n \in \Omega \) be a discrete-state discrete-time homogeneous Markov chain with transition probabilities \( P_{i,j} \) and the following additional properties

- \( \Omega \) is a finite set
- The Markov chain is irreducible
- The Markov chain is aperiodic

Then there exists a unique stationary distribution \( \pi \), which for all states \( i \) is given by

\[
\pi_j = \lim_{n \to \infty} [P^n]_{i,j} > 0 ,
\]

and which is the unique solution to the following set of equations.

\[
1 = \sum_{i \in \Omega} \pi_i \quad (13.25)
\]

\[
\pi_j = \sum_{i \in \Omega} \pi_i P_{i,j} \quad (13.26)
\]

The relations of (13.26) are called the **full-balance equations**. Any probability density which solves the full-balance equations is guaranteed to be the stationary distribution of the Markov chain. Furthermore, in the limit as \( n \to \infty \), the Markov chain is guaranteed to converge to this stationary distribution.
distribution independently of the initial state. Markov chains that have this property of (13.24) are said to be **ergodic**. It can be shown that for ergodic Markov chains, expectations of state variables can be replaced by time averages, which will be very useful in later sections.

### 13.5 Properties of Markov Chains

Theorem [13.4.1] provides relatively simple conditions to establish that a Markov chain has a stationary distribution. However, while it may be known that a stationary distribution exists, it may be very difficult to compute the solution of the full-balance equations in order to determine the precise form of that distribution. Fortunately, Markov chains have a variety of properties that can be extremely useful in both quantifying and understanding their convergence. The following section presents a sampling of some of the most useful properties.

First we derive some basic properties of Markov chains. Our first property is that the conditional distribution of \( X_n \) given some past set of samples, \( X_r \) for \( r < k \), is only dependent on the most recent sample, \( X_k \). Stated formally, we have the following.

**Property 13.1. Conditioning on past of a Markov Chain** - Let \( \{X_n\}_{n=0}^{\infty} \) be a Markov chain taking values on \( \Omega \), and let \( k < n \) be two integers. Then

\[
P\{X_n = x_n | X_r \leq k\} = P\{X_n = x_n | X_k\}.
\]

Intuitively, the present sample is only dependent on the most recent sample that is available in the Markov chain.

This property can be used recursively to show that any subsampling of a Markov chain must also be a Markov chain. So for example, if \( Y_n = X_{dn} \) where \( d \) is a positive integer, then we have that

\[
P\{Y_n = y_n | Y_r < n\} = P\{X_{dn} = x_{dn} | X_{dr} < n\}
\]

\[
= E[P\{X_{dn} = x_{dn} | X_r \leq d(n-1)\} | X_{dr} < n]
\]

\[
= E[P\{X_{dn} = x_{dn} | X_{d(n-1)}\} | X_{dr} < n]
\]

\[
= P\{X_{dn} = x_{dn} | X_{d(n-1)}\}
\]

\[
= P\{Y_n = y_n | Y_{n-1}\},
\]
where the second equality uses the filtration property listed in Property 2.3, the third equality uses Property 13.1 above, and the forth equality uses the result of equation (2.2).

This leads to our next property of Markov chains.

**Property 13.2.** Subsampling of a Markov chain - Let \( \{X_n\}_{n=0}^{\infty} \) be a Markov chain taking values on \( \Omega \), and let \( Y_n = X_{dn} \) where \( d \) is a positive integer. Then \( Y_n \) is a Markov chain. Furthermore, if \( X_n \) is a homogeneous Markov chain, then \( Y_n \) is also a homogeneous Markov chain.

Perhaps surprisingly, it turns out that the concepts of reversibility are often extremely useful in determining the stationary distributions of ergodic Markov chains. So we will need to derive some important conditions related to reversible Markov chains.

Our first observation is that the time-reverse of any Markov chain results in another process with the Markov property.

**Property 13.3.** Time reverse of a Markov chain - Let \( \{X_n\}_{n=0}^{N} \) be a Markov chain taking values on \( \Omega \), and let \( Y_n = X_{N-n} \) for \( n = 0, \cdots, N \). Then \( Y_n \) is a Markov chain.

Perhaps the result of Property 13.3 is surprising given the causal definition of a Markov chain. But in fact, there is hidden symmetry in the definition that arises through the ability to reverse conditional dependence using Bayes’ rule.

It turns out that Property 13.3 provides a powerful tool for us to analyze the convergence of a class of Markov chains. In order to see this, consider the case when \( X_n \) is a homogeneous Markov chain which is in steady state with the stationary distribution \( \pi \). In general, we can write the joint distribution of \( (X_n, X_{n-1}) \) as

\[
P\{X_n = j, X_{n-1} = i\} = P\{X_n = j|X_{n-1} = i\}P\{X_{n-1} = i\} = P\{X_{n-1} = i|X_n = j\}P\{X_n = j\} .
\]

So using this equality, and the definitions of \( \pi_i \) and \( P_{i,j} \), we have that

\[
\pi_i P_{i,j} = P\{X_{n-1} = i|X_n = j\} \pi_j . 
\]

(13.27)

Now we know that the time-reversed process is also a Markov chain, so let
its transition probabilities be denoted by

\[ Q_{j,i} \triangleq P\{X_{n-1} = i|X_n = j\} . \]

Also notice that from the form of equation (13.27), \( Q_{j,i} \) is not a function of \( n \), so the time-reversed Markov chain must also be homogeneous. Plugging back into equation (13.27), we get an expression which must hold for any homogeneous Markov chain in steady state.

\[ \pi_i P_{i,j} = \pi_j Q_{j,i} . \tag{13.28} \]

Now, if in addition, the Markov chain is reversible, then we know that the transitions probabilities of the process running forward in time must be the same as the transition probabilities running backwards in time. Formally, this means that \( P_{i,j} = Q_{i,j} \). Substituting this equality into equation (13.28) yields the so-called detailed-balance equations.

\[ \pi_i P_{i,j} = \pi_j P_{j,i} \tag{13.29} \]

The detailed-balance equations specify that the rate of transitions from state \( i \) to state \( j \) equals the rate of transitions from state \( j \) to \( i \). This is always the case when a Markov chain is reversible and in steady state. This result leads to the following very important definition that extends to both homogeneous and nonhomogeneous Markov chains.

**Definition 18. Reversible Markov Chain**

A Markov chain with transition probabilities \( P\{X_n = j|X_{n-1} = i\} \) is said to be reversible under the stationary distribution \( \pi_i \) if the following detailed-balance equations hold.

\[ P\{X_n = j|X_{n-1} = i\} \pi_i = P\{X_n = i|X_{n-1} = j\} \pi_j \tag{13.30} \]

It is not always possible to find a probability density \( \pi_i \) which is a solution to the detailed-balance equations because not all Markov chains are reversible. But when you can find a solution, then that solution must also be a solution to the full-balance equations, which means that this is the stationary distribution of an ergodic Markov chain!

To see why this is true, we simply sum the detailed-balance equations over
the index $i$.

\[
\sum_{i \in \Omega} P\{X_n = j | X_{n-1} = i\} \pi_i = \sum_{i \in \Omega} P\{X_n = i | X_{n-1} = j\} \pi_j
\]

\[
= \pi_j \sum_{i \in \Omega} P\{X_n = i | X_{n-1} = j\}
\]

\[
= \pi_j
\]

---

**Example 13.3.** A **birth-death process** is a classic example of a reversible Markov chain. In a birth-death process the state of the Markov chain, $X_n$, takes on values in the set $\{0, 1, 2, \cdots \}$. With each new time increment, the value of $X_n$ is either incremented (i.e., birth occurs), decremented (i.e., death occurs), or the state remains unchanged. Mathematically, this can be expressed with the following transition probabilities.

\[
P_{i,j} = \begin{cases} 
\lambda & \text{if } j = i + 1 \\
\mu & \text{if } j = i - 1 \text{ and } i > 0 \\
1 - \lambda - \mu & \text{if } j = i \text{ and } i > 0 \\
1 - \lambda & \text{if } j = i \text{ and } i = 0 \\
0 & \text{otherwise}
\end{cases}
\]

Here $0 < \lambda < 1$ is the probability of a birth and $0 < \mu < 1$ is the probability of a death, where we also assume that $\lambda + \mu < 1$.

In this case, the detailed balance equations have a simple solution. Let $\pi_i$ be the steady-state probability that the Markov chain is in state $i$. Then the detailed balance equations require that

\[
\pi_i \lambda = \pi_{i+1} \mu ,
\]

for all $i \geq 0$. This relations implies the recursion that $\pi_{i+1} = \frac{\lambda}{\mu} \pi_i$ must hold; so we know that the solution must have the form

\[
\pi_i = \frac{1}{z} \left( \frac{\lambda}{\mu} \right)^i ,
\]

where $z$ is a normalizing constant. When $\frac{\lambda}{\mu} < 1$, then the value of $z$ may then be calculated as

\[
z = \sum_{i=0}^{\infty} \left( \frac{\lambda}{\mu} \right)^i = \frac{1}{1 - \frac{\lambda}{\mu}} .
\]
So the stationary distribution is given by

$$\pi_i = \left( \frac{\lambda}{\mu} \right)^i \left( 1 - \frac{\lambda}{\mu} \right).$$

(13.31)

Since the distribution of equation (13.31) solves the detailed balance equations, it must also solve the full balance equations. This implies that it is a stationary distribution of the Markov chain, and that the Markov chain is ergodic and reversible. However, when $\frac{\lambda}{\mu} \geq 1$, then the only solution to the detailed balance equations has the form $\pi_i = 0$. In this case, the Markov chain is not ergodic, which is to say that it does not reach a steady state. Intuitively, when $\lambda \geq \mu$, then the rate of births is greater than the rate of deaths, and the state grows toward infinity, so it can not reach a steady state.

Finally, it is useful to study the convergence behavior of Markov chains with a finite number of states. In this case, the transition probabilities, $P_{i,j}$, can be represented by an $M \times M$ matrix, $P$. In order to further simplify the analysis, let us also assume that the eigenvalues of $P$ are all distinct. In this case, $P$ may be diagonalized using eigen decomposition and expressed in the form

$$P = E^{-1} \Lambda E$$

where the rows of $E$ are the left hand eigenvectors of $P$, and $\Lambda$ is a diagonal matrix of eigenvalues. Using this decomposition, we can see that

$$P^n = P^{n-2} E^{-1} \Lambda E E^{-1} \Lambda E$$
$$= P^{n-2} E^{-1} \Lambda^2 E$$
$$= E^{-1} \Lambda^n E.$$

So the distribution at time $n$ is given by

$$\pi^{(n)} = \pi^{(0)} E^{-1} \Lambda^n E.$$

Then we can write this in terms of its modal components as

$$\pi^{(n)} = \sum_{m=0}^{M-1} c_m \Lambda_m^n E_{m,*}.$$
where $E_{m,*}$ is the $m^{th}$ row of $E$, and $c_m$ is the $m^{th}$ component of a constant vector given by $c = \pi(0)E^{-1}$. When $P$ corresponds to an irreducible and aperiodic Markov chain, then it must have a stationary distribution. In this case, exactly one of the eigenvalues is 1, and the remaining eigenvalues have magnitude strictly less than 1. In this form, we can see that all the other modes with eigenvalues $|\Lambda_{m,m}| < 1$ must decay away at a geometric rate. So, in principle, convergence of the Markov chain to its stationary distribution should be rapid.
13.6 Chapter Problems

1. Let \( \{X_i\}_{i=0}^N \) be a Markov Chain with \( X_i \in \{0, \ldots, M-1\} \) and transition probabilities given by \( P\{X_n = j|X_{n-1} = i\} = P_{i,j} \) where \( 0 \leq i, j < M \), and initial probability \( P\{X_0 = j\} = \tau_j \) and parameters \( \theta = (\tau, P) \). Also define the statistics

\[
N_j = \delta(X_0 - j) \quad \text{and} \quad K_{i,j} = \sum_{n=1}^{N} \delta(X_n - j)\delta(X_{n-1} - i).
\]

a) Use these statistics to derive an expression for the probability \( p(x) \).

b) Show that the Markov chain has an exponential distribution, and that \( N_j \) and \( K_{i,j} \) are the natural sufficient statistics of the distribution.

c) Derive the ML estimate of the parameters \( \tau_j \) and \( P_{i,j} \) in terms of the natural sufficient statistics.

2. Use the assumed properties of the HMM to prove the equality of equation (13.6).

3. Show that the recursions of equations (13.7), (13.8), and (13.9) are correct.

4. Use the definitions of \( \alpha_n(i) \) and \( \beta_n(i) \) to derive the relationships of equations (13.10), (13.11), (13.12), and (13.13).

5. Derive the recursions of the forward-backward algorithm given in equations (13.14) and (13.15).

6. Consider the HMM, \((X, Y)\), with the structure described in Section (13.3), but with the observed quantity \( Y_n \) being a \( p \) dimensional multivariate Gaussian random vector with conditional distribution \( N(\mu_{x_n}, R_{x_n}) \). Show that the following are natural sufficient statistics for the joint
distribution, $p(x, y|\theta)$, with parameters $\theta = [\mu_j, R_j, \tau_j, P_{i,j} : \text{for } i, j \in \Omega]$.

\[
\begin{align*}
b_j &= \sum_{n=1}^{N} y_n \delta(x_n - j) \\
S_j &= \sum_{n=1}^{N} y_n y_n^t \delta(x_n - j) \\
N_j &= \delta(x_0 - j) \\
K_{i,j} &= \sum_{n=1}^{N} \delta(x_n - j) \delta(x_{n-1} - i).
\end{align*}
\]

7. Prove Property [13.1]

8. Prove Property [13.2]

9. Property [13.3]

10. Let $X_n$ be a Markov chain (not necessarily homogeneous) that takes values on the discrete set $\Omega$. And let $Y_n = X_{Dn}$ be a subsampled version of $X_n$ where $D$ is a positive integer.

a) Show that $Y_n$ is also a Markov chain.

b) Show that if $X_n$ is a homogeneous Markov chain, then $Y_n$ is a homogeneous Markov chain.

c) Show that if $X_n$ is a reversible Markov chain, then $Y_n$ is a reversible Markov chain.

11. Consider the homogeneous Markov chain $\{X_n\}_{n=0}^{\infty}$ with parameters

\[
\begin{align*}
\tau_j &= P\{X_0 = j\} \\
P_{i,j} &= P\{X_n = j|X_{n-1} = i\}
\end{align*}
\]

where $i, j \in \{0, \cdots, M-1\}$. Furthermore, assume that the transition parameters are given by

\[
P_{i,j} = \begin{cases} 
1/2 & \text{if } j = (i + 1) \text{mod} M \\
1/2 & \text{if } j = i \\
0 & \text{otherwise}
\end{cases}
\]
a) Write out the transition matrix $P$ for the special case of $M = 4$. (But solve the remaining problems for any $M$.)

b) Is the Markov chain irreducible? Prove or give a counter example.

c) Is the Markov chain periodic? Prove or give a counter example.

d) Is the Markov chain ergodic? Prove or give a counter example.

e) Determine the value of the following matrix

$$\lim_{n \to \infty} P^n$$

f) Is the Markov chain reversible? Prove or give a counter example.

12. Consider a Markov chain, $X_n$, as in Example [13.3], but with the states limited to the set \{0, \ldots, M-1\}. So for $i = 0, \ldots, M-2$, then $P_{i,i+1} = \lambda$ is the birth rate; and for $i = 1, \ldots, M-1$, then $P_{i,i-1} = \mu$ is the death rate. Calculate all the values of $P_{i,j}$ and the density $\pi_i$ that is the stationary distribution of the Markov chain.
Chapter 14

General MRF Models

In the previous Chapter, we showed how state sequence estimation of HMMs can be used to label samples in a 1-D signal. In 2-D, this same approach corresponds to image segmentation, i.e. the labeling of pixels into discrete classes. However, in order to formalize this approach to image segmentation, we will first need to generalize the concept of a Markov Chain to two or more dimensions. As with the case of AR models, this generalization is not so straightforward because it implies the need for non-causal prediction, which results in loopy dependencies in the model. Generally, these loopy dependencies make closed form calculation of the probability mass function impossible due to intractable form of the so-called partition function, or normalizing constant of the distribution.

A common partial solution to this dilemma is to use discrete Markov random field (MRF) models. As with the non-Gaussian MRFs of Chapter 6, the discrete MRFs do not require imposing a 1-D causal ordering of the pixels in the plane, and therefore they can produce more natural and isotropic image models. However, as we will see, a disadvantage of MRF models is that parameter estimation can become much more difficult due to the intractable nature of the partition function. The key theorem required to work around this limitation is the Hammersley-Clifford Theorem which will be presented in detail.
### 14.1 MRFs and Gibbs Distributions

Before we can define an MRF, we must first define the concept of a neighborhood system. Let $S$ be a set of lattice points with elements $s \in S$. Then we use the notation $\partial s$ to denote the neighbors of $s$. Notice that $\partial s$ is a subset of $S$, so the function $\partial$ is a mapping from $S$ to the power set of $S$, or equivalently the set of all subsets of $S$ denoted by $2^S$.

However, not any mapping $\partial s$ qualifies as a neighborhood system. In order for $\partial s$ to be a neighborhood system, it must meet the following symmetry constraint.

**Definition 19. Neighborhood system**

*Let $S$ be a set of lattice of points, then any mapping $\partial : S \rightarrow 2^S$ is a neighborhood system if for all $s, r \in S$*

$$ r \in \partial s \Rightarrow s \in \partial r \quad \text{and} \quad s \notin \partial s $$

In other words, if $r$ is a neighbor of $s$, then $s$ must be a neighbor of $r$; and in addition, $s$ may not be a neighbor of itself. Notice that this definition is not restricted to a regular lattice. However, if the lattice $S$ is a regular lattice, and the neighborhood is spatially invariant, then symmetry constraint necessitates that the neighbors of a point must be symmetrically distributed about each pixel. Figure 14.1a shows such a symmetric 8-point neighborhood.

We may now give the general definition of an MRF.
Definition 20. Markov Random Field
Let $X_s \in \Omega$ be a discrete (continuous) valued random field defined on the lattice $S$ with neighborhood system $\partial s$. Further assume that the $X$ has probability mass (density) function $p(x)$. Then we say that $X$ is a Markov random field (MRF) if it has the property that for all $x \in \Omega$

$$P\{X_s = x_s | X_r \text{ for } r \neq s\} = P\{X_s = x_s | X_{\partial r}\},$$

for all $x_s \in \Omega$.

So each pixel of the MRF is only dependent on its neighbors.

A limitation of MRFs is that their definition does not yield a natural method for writing down an MRFs distribution. At times, people have been tempted to write the joint distribution of the MRF as the product of the associated conditional distributions; however, this is generally not true. More formally, it is typically the case that

$$p(x) \neq \prod_{s \in S} p(x_s | x_{\partial r})$$

because the loopy dependencies of a MRFs non-causal structure render the result of (13.1) invalid.

Alas, this limitation of MRFs is quite fundamental, but the good news is that many researchers have been provided gainful employment in an attempt to solve it. Perhaps the most widely used work-around to this dilemma has been the Gibbs distribution. However, in order to define the Gibbs distribution, we must first introduce the concept of cliques defined below.

Definition 21. Clique
Given a lattice $S$ and neighborhood system $\partial s$, a clique is any set of lattice points $c \subset S$ such that for every distinct pair of points, $s, r \in c$, it is the case that $r \in \partial s$.

Cliquess are sets of point which are all neighbors of one another. Examples of cliques for an eight point neighborhood system on a rectangular lattice are illustrated in Figure [14.1b]. Notice, that any individual point, $s \in S$, forms a clique with the single member, $c = \{s\}$.

1 In the continuous case, this equality becomes $P\{X_s \in A | X_r \text{ for } r \neq s\} = P\{X_s \in A | X_{\partial r}\}$, for all measurable sets $A \subset \Omega$, and in both cases the equality is with probability 1.
In fact, cliques and neighborhood systems are alternative specifications of the same information because for each neighborhood system, there is a unique corresponding maximal set of cliques, and for each set of cliques there is a unique corresponding minimal neighborhood system. (See problem 1)

With this definition of cliques, we may now define the concept of a Gibbs distribution.

**Definition 22. Gibbs Distribution**

Let \( p(x) \) be the probability mass (density) function of a discrete (continuous) valued random field \( X_s \in \Omega \) defined on the lattice \( S \) with neighborhood system \( \partial S \). Then we say that \( p(x) \) is a Gibbs distribution if it can be written in the form

\[
p(x) = \frac{1}{z} \exp \left\{ - \sum_{c \in \mathcal{C}} V_c(x_c) \right\}
\]

(14.1)

where \( z \) is a normalizing constant known as the partition function, \( \mathcal{C} \) is the set of all cliques, \( x_c \) is the vector containing values of \( x \) on the set \( c \), and \( V_c(x_c) \) is any functions of \( x_c \).

We sometimes refer to the function \( V_c(x_c) \) as a potential function and the function

\[
u(x) = \sum_{c \in \mathcal{C}} V_c(x_c)
\]

as the associated energy function.

The important result that relates MRFs and Gibbs distributions is the Hammersley-Clifford Theorem\(^6\) stated below.

**Theorem 14.1.1. Hammersley-Clifford Theorem**

Let \( S \) be an \( N \) point lattice with neighborhood system \( \partial S \), and \( X \) be a discrete (continuously) valued random process on \( S \) with strictly positive probability mass (density) function \( p(x) > 0 \). Then \( X \) is an MRF if and only if \( p(x) \) is a Gibbs distribution.

So the Hammersley-Clifford Theorem states that \( X \) is an MRF if and only if its density function, \( p(x) \), is a Gibbs distribution. Since, Gibbs distributions are restricted to be strictly positive, this result comes along with the technical constraint that \( X \) has a strictly positive density, but in practice this is a minor limitation since the density can become exceedingly small even if it is non-zero.
Interestingly, the proof that a Gibbs distribution corresponds to an MRF is simple and constructive, while the converse proof that an MRF must have a Gibbs distribution, is not.\footnote{The proof that every MRF has a Gibbs distribution depends on eliminating the possibility that the \( \log p(x) \) depends on functions of non-cliques.} In practice, this means that one typically starts with a Gibbs distribution, and from this computes the conditional dependencies of an MRF, and not the other way around.

So to prove that a Gibbs distribution corresponds to an MRF, we first define \( \mathcal{C}_s \) to be the set of all cliques that contain the pixel \( x_s \). More precisely, let \( \mathcal{C}_s \) be a subset of all cliques, \( \mathcal{C} \), defined by

\[
\mathcal{C}_s = \{ c \in \mathcal{C} : s \in c \},
\]

and let \( \bar{\mathcal{C}}_s = \mathcal{C} - \mathcal{C}_s \) be the set of all cliques that do not include \( x_s \). The we can explicitly compute the conditional distribution of \( x_s \) given all other pixels as

\[
P\{ X_s = x_s | X_r \neq s \} = \frac{\frac{1}{z} \exp \left\{ - \sum_{c \in \mathcal{C}} V_c(x_c) \right\}}{\sum_{x_s \in \Omega} \frac{1}{z} \exp \left\{ - \sum_{c \in \mathcal{C}} V_c(x_c) \right\}}
\]

\[
= \frac{\exp \left\{ - \sum_{c \in \mathcal{C}_s} V_c(x_c) \right\} \exp \left\{ - \sum_{c \in \bar{\mathcal{C}}_s} V_c(x_c) \right\}}{\sum_{x_s \in \Omega} \exp \left\{ - \sum_{c \in \mathcal{C}_s} V_c(x_c) \right\} \exp \left\{ - \sum_{c \in \bar{\mathcal{C}}_s} V_c(x_c) \right\}}
\]

\[
= \frac{\exp \left\{ - \sum_{c \in \mathcal{C}_s} V_c(x_c) \right\}}{\sum_{x_s \in \Omega} \exp \left\{ - \sum_{c \in \mathcal{C}_s} V_c(x_c) \right\}}
\]

\[
= \exp \left\{ - u_s(x_s|\partial s) \right\}
\]

where

\[
u_s(x_s|\partial s) = \sum_{c \in \mathcal{C}_s} V_c(x_c)
\]

is only a function of \( x_s \) and its neighbors. So from this, we see that

\[
P\{ X_s = x_s | X_r \neq s \} = P\{ X_s = x_s | X_r \in \partial s \},
\]

and \( X_s \) must be an MRF.

The Hammersley-Clifford Theorem also justifies the pair-wise Gibbs distribution approach taken in Chapter 6 since neighboring pixel pairs represent
cliques in any neighborhood system. However, referring to Figure [14.1] it’s interesting to note that even for an 8-point neighborhood system other cliques exist; so pair-wise Gibbs distributions are certainly not the most general class of MRFs possible for a given neighborhood system. Nonetheless, the pair-wise Gibbs distributions admit a rich and useful sub-class of MRFs.

Now since most approaches start with a Gibbs distribution and then derive the associated MRF, it is important to understand how a set of cliques can induce a neighborhood system. So let’s say that \( X_s \) has a Gibbs distribution on the lattice \( S \) with cliques \( C \). Then it must be an MRF, but what is the smallest neighborhood system, \( \partial s \), that is guaranteed to fully describe that MRF? Formally, the answer to this question is given by the following minimal neighborhood system.

\[
\partial s = \{ r \in S : \exists c \in C \text{ s.t. } r \neq s \text{ and } r, s \in c \} \quad (14.4)
\]

So in words, the minimal neighborhood, \( \partial s \), must contain any pixel, \( r \), that shares a clique with \( s \); and it may not contain any more pixels. Perhaps this will become clearer when we need to use this fact in a number of later examples.

An important limitation of the Gibbs distribution is that typically, it is impossible to compute or even express in simple closed form the partition function \( z \). Theoretically, \( z \) can be written as the sum over all possible states of the random field \( X \),

\[
z = \sum_{x \in \Omega^{|S|}} \exp \{-u(x)\} ,
\]

where each point in the lattice, \( x_s \), takes on values in the discrete set \( \Omega \). However, in practice, this expression can not be evaluated since it requires the summation over an exponentially large set of states. This becomes a serious limitation in problems such as parameter estimation in which it is
often necessary to maximize or at least compute the likelihood as a function of a parameter. So for example, in order to compute the ML estimate of a parameter \( \theta \), it is necessary to minimize the expression,

\[
\hat{\theta} = \arg \min_{\theta} \{ u(x) + \log z(\theta) \} ;
\]

however, this is generally not possible to compute in closed form when the evaluation of the partition function is intractable.

Nonetheless, the Gibbs distribution is still extremely valuable when the problem is one of estimating or maximizing over \( x \). Since the partition function is not a function of \( x \), optimization with respect to \( x \) can be performed using only the energy function, without regard to the potentially intractable partition function.

\[
\hat{x} = \arg \max_{x} \{ \log p(x) \} = \arg \min_{x} \{ u(x) \} .
\]

Another case of great importance occurs when we would like to sample from the distribution of a Gibbs distribution. This problem, which is considered in detail in Chapter 15, generally only requires the computation of the likelihood ratio between alternative states, say \( x \) and \( x' \). In this case, the likelihood ratio can be exactly computed from the Gibbs distribution

\[
\frac{p(x')}{p(x)} = \exp \{ u(x) - u(x') \} .
\]

### 14.2 1-D MRFs and Markov Chains

In order to better understand the value of MRFs, it is useful to start with the 1-D case. So consider a 1-D random process, \( \{X_n\}_{n=0}^N \), which takes values on the discrete set \( \{0, \cdots, M - 1\} \).

First, it is easily shown that if \( X_n \) is a Markov chain, then it must also have a Gibbs distribution, which further implies that it must be an MRF. To see this, we may write the PMF of the Markov chain as,

\[
p(x) = p_0(x_0) \prod_{n=1}^N p_n(x_n|x_{n-1})
\]

\[
= \exp \left\{ \sum_{n=1}^N \log p_n(x_n|x_{n-1}) + \log p_0(x_0) \right\} ,
\]
where \( p_n(x_n|x_{n-1}) \) is the transition distribution for the Markov chain, and \( p_0(x_0) \) is the PMF of the initial state.\(^3\) If we take the potential functions to be

\[
V_n(x_n, x_{n-1}) = -\log p_n(x_n|x_{n-1}),
\]

for \( 2 \leq n \leq N \), and

\[
V_1(x_n, x_{n-1}) = -\log p_1(x_1|x_0) - \log p_0(x_0).
\]

Then with this definition, we see that the distribution of a Markov chain is a Gibbs distribution with the form

\[
p(x) = \frac{1}{z} \exp \left\{ -\sum_{n=1}^{N} V_n(x_n, x_{n-1}) \right\},
\]

where the cliques are formed by \( c_n = \{n, n-1\} \), and the full set of cliques is given by \( C = \{c_n : \text{for } n = 1, \cdots N\} \).

Since \( X_n \) has a Gibbs distribution, then by the Hammersley-Clifford Theorem, \( X_n \) must be an MRF. Furthermore, its neighborhood system must be specified by the result of equation (14.4). This means that \( \partial n \) must include any pixel that shares a clique with \( n \). For this simple case, the neighbors are formally specified to be

\[
\partial n = \{n - 1, n + 1\} \cap \{0, \cdots, N\}.
\]

So the neighbors of \( n \) will be the pixels on either side of \( n \), except when \( n \) falls on a boundary, in which case it will only have a single neighbor.

In fact, we can generalize these ideas to a \( P^{th} \) order Markov chain. In this case, each sample is only dependent on the \( P^{th} \) previous samples as defined below.

**Definition 23.** \( P^{th} \) Order Markov Chain

\( \{X_n\}_{n=0}^{N} \) is said to be a \( P^{th} \) order Markov chain if for all \( n \), the following holds.

\[
P\{X_n = x_n|X_r = x_r \text{ for } r < n\} = P\{X_n = x_n|X_{n-1}, \cdots, X_{\max\{n-P,0\}}\}
\]

\(^3\)Here we again assume that the probabilities are non-zero, so as to meet the technical constraint of the Hammersley-Clifford Theorem, but this is not a strong restriction since these probabilities can be allowed to be very close to zero.
Since a $P^{th}$ order Markov chain is only conditionally dependent on the last $P$ values of the sequence, we can write the joint distribution of the sequence as the product of the transition probabilities.

$$p(x) = p_0(x_0, \cdots, x_{P-1}) \prod_{n=P}^{N} p_n(x_n|x_{n-1}, \cdots, x_{n-P})$$

Again, by taking the log, we can write this in the form

$$p(x) = \frac{1}{z} \exp \left\{ - \sum_{n=P}^{N} V_n(x_n, \cdots, x_{n-P}) \right\} . \quad (14.5)$$

So the cliques become $c_n = \{n, \cdots, n-P\}$, and the full set of cliques is given by $\mathcal{C} = \{c_n : n = 1, \cdots, N\}$.

Once again we see that the expression of equation (14.5) is a Gibbs distribution, so $X_n$ must be an MRF; and the neighborhood system of $X_n$ must be given by $\partial n = \{n-P, \cdots, n-1, n+1, \cdots, n+P\} \cap \{0, \cdots, N\}$. (See problem 4) Therefore, in 1-D, we can see that a $P^{th}$ order Markov chain is always a $P^{th}$ order MRF, as defined below.

**Definition 24.** $P^{th}$ Order 1-D Markov Random Field

\{X_n\}_{n=0}^{N} is said to be a $P^{th}$ order 1-D Markov random field if for all $n$, the following holds.

$$P\{X_n = x_n|X_r = x_r \text{ for } r \neq n\} = P\{X_n = x_n|X_{\max\{n-P,0\}}, \cdots, X_{n-1}, X_{n+1}, \cdots, X_{\min\{n+P,N\}}\}$$

So if a 1-D Markov chain is an MRF, is the converse always true? In order to answer this question, consider when $\{X_n\}_{n=0}^{N}$ is an order $P = 1$ MRF in 1-D. In this case, the MRF must have a Gibbs distribution with a neighborhood system given by $\partial n = \{n-1, n+1\} \cap \{0, \cdots, N\}$. With this neighborhood system, the most general set of cliques is given by $\mathcal{C} = \{\{n, n-1\} : n = 1, \cdots, N\}$\footnote{Of course, any individual sample, $x_n$, is also a clique, but, this clique’s potential function, $V'_n(x_n)$, can always be subsumed as part of the potential function, $V_n(x_n, x_{n-1})$.} So therefore, we know that it must be possible to write the Gibbs distribution with the form

$$p(x) = \frac{1}{z} \exp \left\{ - \sum_{n=1}^{N} V_n(x_n, x_{n-1}) \right\} .$$
From this, it is easy to show that the truncated random process, from time 
\( n = 0 \) to \( N - 1 \) is also an MRF. We can do this by showing it also has a 
Gibbs distribution given by

\[
p(x_0, \cdots, x_{N-1}) = \sum_{x_N=0}^{M-1} \frac{1}{z} \exp \left\{ - \sum_{n=1}^{N} V_n(x_n, x_{n-1}) \right\}
\]

\[
= \frac{1}{z} \exp \left\{ - \sum_{n=1}^{N-1} V_n(x_n, x_{n-1}) \right\} \sum_{x_N=0}^{M-1} \exp \left\{ -V_N(x_N, x_{N-1}) \right\}
\]

\[
= \frac{1}{z} \exp \left\{ - \sum_{n=1}^{N-1} V_n(x_n, x_{n-1}) \right\} \exp \left\{ -V'_{N-1}(x_{N-1}) \right\}
\]

\[
= \frac{1}{z} \exp \left\{ - \sum_{n=1}^{N-1} V_n(x_n, x_{n-1}) - V'_{N-1}(x_{N-1}) \right\}
\]

where all but the last of the potential functions are the same.

Using this result, we can also compute the conditional PMF of \( X_n \) given 
\( X_r \) for \( r < n \) as,

\[
p(x_N | x_r \text{ for } r < N) = \frac{p(x_0, \cdots, x_N)}{p(x_0, \cdots, x_{N-1})}
\]

\[
= \frac{\frac{1}{z} \exp \left\{ - \sum_{n=1}^{N} V_n(x_n, x_{n-1}) \right\}}{\frac{1}{z} \exp \left\{ - \sum_{n=1}^{N-1} V_n(x_n, x_{n-1}) - V'_{N-1}(x_{N-1}) \right\}}
\]

\[
= \exp \left\{ -V_N(x_N, x_{N-1}) + V'_{N-1}(x_{N-1}) \right\}
\]

\[
= \frac{1}{z'} \exp \left\{ -V_N(x_N, x_{N-1}) \right\}
\]

\[
= p(x_N | x_{N-1}) ,
\]

where \( z' = \sum_{x_N=0}^{M-1} \exp \left\{ -V_N(x_N, x_{N-1}) \right\} \).

So applying this result, along with mathematical induction, we see that 
every 1-D MRF of order 1 is also a 1-D Markov chain of order 1. In fact,
the generalization of this proof to \( P^{th} \) order 1-D MRFs and Markov chains is direct (but with nasty notional issues). This results in the following very important property for 1-D random processes.

**Property 14.1. Equivalence of 1-D MRFs and Markov chains** - Let \( \{X_n\}_{n=0}^N \) be discrete time random process with strictly positive PMF taking values on a finite discrete set. Then \( X \) is a \( P^{th} \) order MRF if and only if it is a \( P^{th} \) order Markov chain.

Of course, the sums in the proof can easily be replaced with integrals, so a similar result holds for discrete-time processes with strictly positive densities. That is, \( X_n \) is a \( P^{th} \) order 1-D discrete-time Markov process with a strictly positive PDF, if and only if it is \( P^{th} \) order 1-D MRF with a strictly positive PDF.

In some sense, this would seem to be a major disappointment. If MRFs and Markov chains are equivalent, why bother with MRFs at all? Well the answer is that MRFs are rarely used in 1-D for this exact reason. However, in the following sections, we will see that in two a more dimensions, MRFs are quite different than 2-D Markov processes, and can exhibit interesting and useful characteristics.

### 14.3 Discrete MRFs and the Ising Model

Perhaps the first serious attempts to generalize Markov chains to multiple dimensions came from the physics community in an effort to model the behavior and properties of magnetic dipoles in real materials. Amazingly enough, some of this early research resulted in closed form expressions for the behavior of these highly non-linear systems, but they achieved this goal by restricting their analysis to simple 2-D models with 4-point neighborhoods.

In the 1920’s, it was well known that magnetic materials exhibit a phase transition when the temperature falls below the **Curie temperature**. Below this **critical temperature**, the magnetic material will spontaneously magnetize itself, but above the critical temperature it will not. Wilhelm Lenz [43] and his student, Ernst Ising [39], proposed what is now known as the **Ising model** in an attempt to mathematically model this unique physical behavior.
The Ising model is formed by a lattice of binary random variables that are used to model the states of magnetic dipoles. Each binary variable, \( X_s \), can take on the value +1 or −1, indicating either a North or South orientation of the corresponding dipole. The potential energy associated with a pair of neighboring dipoles is then approximated as \(-J x_s x_r\) where \( s \in \partial r \) and \( J \) is a positive constant. So if the neighboring dipoles are misaligned, (i.e., \( x_s = -x_r \)), then the potential energy is \( J \); and if they are aligned, (i.e., \( x_s = x_r \)), then the potential energy falls to \(-J\). Using this model, the total energy of the lattice is given by

\[
u(x) = - \sum_{\{s,r\} \in C} J x_s x_r,
\]

where \( C \) is the set of all neighboring pixel pairs (i.e., cliques).

The question that next arises is, what is the distribution of the dipole states in the Ising model? For a physical system, the question is answered by the Boltzmann distribution (also known as the Gibbs distribution) of thermodynamics. The Boltzmann distribution has the form

\[
p(x) = \frac{1}{z} \exp \left\{ - \frac{1}{k_b T} \nu(x) \right\},
\]

where \( \nu(x) \) is the energy associated with the state \( x \). The Boltzmann distribution is derived for a system in thermodynamic equilibrium by finding the distribution of states that maximizes the entropy of the system subject to a constraint that the expected energy be constant. (See Appendix B and problem 6 for a derivation of this result.) So for the Ising model, the Boltzmann distribution of states has the form

\[
p(x) = \frac{1}{z} \exp \left\{ \frac{1}{k_b T} \sum_{\{s,r\} \in C} J x_s x_r \right\}.
\]  

(14.6)

Since this is a Gibbs distribution, then we know that \( p(x) \) must be the distribution of an MRF with cliques given by \( C \).

In order to be able to analyze the properties of this model, Ising first assumed the lattice to be only one dimensional. Now we know from Section 14.2 that any 1-D MRF of order \( P \) is a 1-D Markov chain of order \( P \). So this 1-D approximation allowed Ising to exactly analyze the behavior of
the lattice. However, in 1-D, the Ising model no longer exhibits the critical temperature behavior! So let $X_n$ be the 1-D random process resulting from the Ising model, and assume that we know that $X_0 = 1$. Then as $n \to \infty$, we know that the Markov chain will converge to a stationary distribution and $X_n$ will be independent of the initial value, $X_0$. This means that in 1-D the initial state has no lasting effect.

However, in 2-D, Gibbs distributions no longer can be represented as simple Markov chains. So the analysis is much more difficult, but also turns out to yield much richer behavior. In fact in 1936, Peierls first provided a precise argument that in 2-D the Ising model exhibits the expected critical temperature behavior [51]. So below a specific critical temperature, dipoles in the random field remain correlated regardless of how far apart they are. Later this analysis was made more precise when in 1944 Onsager was able to obtain a closed form expression for the infinite lattice case [49].

In order to better understand the Ising model, we first make the standard modeling assumption of a 2-D rectangular lattice with a 4-point neighborhood system, and cliques formed by neighboring horizontal or vertical pixel pairs. If we use the notation, $\delta(x_r \neq x_s)$, as an indicator function for the event $x_r \neq x_s$, then we can rewrite the Ising distribution in the form

$$
p(x) = \frac{1}{z} \exp \left\{ \frac{J}{k_b T} \sum_{\{r,s\} \in \mathcal{C}} x_r x_s \right\}
= \frac{1}{z} \exp \left\{ \frac{J}{k_b T} \sum_{\{r,s\} \in \mathcal{C}} (1 - 2 \delta(x_r \neq x_s)) \right\}.
$$

This results in a new expression for the distribution of an Ising model

$$
p(x) = \frac{1}{z(\beta)} \exp \left\{ -\beta \sum_{\{r,s\} \in \mathcal{C}} \delta(x_r \neq x_s) \right\},
\quad \text{(14.7)}
$$

where

$$
\beta \triangleq \frac{2J}{k_b T}
\quad \text{(14.8)}
$$
is a standard parameter of the Ising model that is inversely proportional to the temperature. Notice that we write $z(\beta)$ to emphasize the dependence of
the partition function on \( \beta \). Generally, it is more convenient and intuitive to work with the unitless parameter of \( \beta \) than to work with the temperature \( T \) directly.

From the form of (14.7), we know that \( p(x) \) is a Gibbs distribution with pair-wise cliques. In 2-D, the Ising model uses horizontal and vertical cliques, so by the Hammersly-Clifford Theorem, we also know that \( X \) must be a MRF with a 4-point neighborhood. More directly, we can compute the conditional distribution of a pixel \( x_s \) given its neighbors \( x_r \) for \( r \in \partial s \) to yield

\[
p(x_s|x_i \neq s) = \frac{\exp \left\{ -\beta \sum_{r \in \partial s} \delta(x_s \neq x_r) \right\}}{\exp \left\{ -\beta \sum_{r \in \partial s} \delta(1 \neq x_r) \right\} + \exp \left\{ -\beta \sum_{r \in \partial s} \delta(-1 \neq x_r) \right\}}.
\]

(14.9)

In order to further simplify this expression, we can define \( v(x_s, \partial x_s) \) to be the number of horizontal/vertical neighbors that differ from \( x_s \).

\[
v(x_s, \partial x_s) \triangleq \sum_{r \in \partial s} \delta(x_s \neq x_r)
\]

(14.10)

From this we can compute the conditional probability that \( x_s = 1 \) as

\[
p(1|x_i \neq s) = \frac{1}{1 + e^{2\beta[v(1,\partial x_s)-2]}}.
\]

(14.11)

Figure 14.3 shows a plot of the function in equation (14.11). First, notice that the conditional probability is only a function of \( v(1, \partial x_s) \), that is the number of neighbors not equal to 1. When \( \beta > 0 \), then as \( v(1, \partial x_s) \) increases, the probability that \( X_s = 1 \) decreases. A special case occurs when \( v(1, \partial x_s) = 2 \). In this case, half of the neighbors are 1, and half are \(-1\), so \( p(x_s|x_{\partial s}) = 1/2 \).

Generally speaking, when \( \beta \) is very large and positive, a pixel is highly likely to be the same polarity as the majority of its neighbors. However, if \( \beta << 0 \), then the opposite occurs, and pixels are likely to be the opposite polarity of their neighbor. In the transitional case, when \( \beta = 0 \), the pixels in the Ising model are all independent. Intuitively, \( \beta = 0 \) corresponds to \( T = \infty \), so at infinite temperature, the dipoles of the lattice are highly disordered with no correlation between them.

Figure 14.4 provides additional intuition into the behavior of the Ising model. Notice that with its 4-point neighborhood, the energy function of the
Ising model is equal to the total boundary length scaled by the constant $\beta$.

$$\beta \sum_{\{r,s\} \in C} \delta(X_r \neq X_s) = \beta(\text{Boundary length})$$

So with this interpretation, we might expect that longer boundaries should become less probable as the value of $\beta$ increases. However, this is not necessarily the case because as the length of a boundary increases the total number of possible boundaries of a given length can increase quite rapidly.

The complex nature of the 2-D Ising model is perhaps best illustrated by the existence of a critical temperature below which the correlations in the lattice extend to infinite distances. Figure 14.5 illustrates an Ising model of size $N \times N$ with a boundary specified by the set of lattice points $B$. So one might expect that as the size of the lattice, $N$, tends to infinity, then the choice of the boundary as either $X_B = 1$ or $X_B = -1$ would not affect the pixels in the center of the lattice. However, this is not true. In 1936, Peierls first showed this surprising result using a clever proof based on arguments about the probability of a boundary surrounding the center pixel [51]. Kinderman and Snell provide a simplified example of this proof technique on page 13 of [42] to show that for $\beta$ sufficiently large the probability of the center pixel, $X_0$, can be made arbitrarily small. For $\beta > \beta_c$, it can be shown...
that
\[
\lim_{N \to \infty} P\{X_0 = -1|X_B = 1\} < 1/2.
\]
In fact, by choosing \(\beta\) sufficiently large, it can be shown that \(\lim_{N \to \infty} P\{X_0 = -1|X_B = 1\} < \epsilon\) for any \(\epsilon > 0\). This means that the choice of a boundary effects the behavior of the Ising model even as the boundary goes to infinity. This is a behavior which is distinctly different than the behavior of an ergodic 1-D Markov chain, in which the effect of the initial value decays with time.

**Example 14.1.** In this example, we provide a simple geometric proof originally from Kinderman and Snell (See [42] page 13) that there exists a critical value, \(\beta_c\), above which correlations in the lattice extend to infinite distances. The proof works by first bounding the correlation for the case of an \(N \times N\) lattice, and then taking the limit as \(N \to \infty\).

Figure 14.5a illustrates the situation in which an \(N \times N\) Ising MRF has a center pixel \(X_0\) which is surrounded by an outer boundary of values \(B\). Our objective will be to show that for \(\beta\) sufficiently large, \(P_N\{X_0 = -1|B = 1\} < 1/3\). Once we have proven this, we than know that
\[
\lim_{N \to \infty} P_N\{X_0 = B\} \geq 2/3,
\]
which means that the value of \(X_0\) and the boundary \(B\) are correlated regardless of how large the lattice becomes. So intuitively, this center pixel of the random field is effected by the boundary regardless of how far away the boundary moves.
In order to formulate the proof, we will first need to define some notation. Consider a particular configuration \( x \) for which \( x_0 = -1 \) and \( B = 1 \). Then we know there must be a closed boundary that surrounds the lattice point at 0 and that contains all the pixels with label \(-1\) that are connected through 4-point neighbor relation to the pixel \( X_0 = -1 \). This boundary and the associated set of pixels inside the boundary are illustrated in Figure 14.5a.

Now let \( \Omega = \{+1, -1\}^{N^2} \) denote the set of all binary images, and let \( L(x) \) be the total boundary length associated with neighboring pixels of opposite polarity. Then the probability distribution of \( X \) given \( B = 1 \) has the form

\[
P\{X = x | X_B = 1\} = \frac{1}{z} \exp \{-\beta L(x)\}.
\]

Now let \( c(x) \) denote the closed boundary around the pixel \( X_0 \) that includes all pixels of value \(-1\). Notice that all pixels inside the curve, \( c(x) \), must have value \(-1\), and all values immediately outside the curve must be \(+1\). In addition, since the outer boundary is assumed to be all 1's (i.e., \( B = 1 \)) and \( X_0 = -1 \), then \( c(x) \) must exist as a separation between the change in these two states. We will denote the length of this separating curve by \( L(c(x)) \). Figure 14.5a illustrates the situation with \( X_0 = -1 \) and \( c(x) \) denoted by a bold line of length \( L(c(x)) = 24 \) enclosing 15 pixels each with the value \(-1\).

Next, consider the set of all fields, \( x \), such that \( x_0 = -1 \), and \( c(x) = c_o \).

\[
\Omega_{c_o} \triangleq \{x \in \Omega : c(x) = c_o \text{ and } x_0 = -1\}.
\]

So then for any \( x \in \Omega_{c_o} \), we know that all the pixels inside the boundary \( c_o \) will have value \(-1\) including \( x_0 \), and all the neighboring pixels on the outside boundary of \( c_o \) will be \(+1\).

Now for each \( x \in \Omega_{c_o} \), we can define a “flipped” version of \( x \) by reversing the polarity of the pixels inside the boundary \( c_o \), but leaving the pixels outside the boundary unchanged. So if \( x' = f_{c_o}(x) \) is the flipped version of \( x \), then

\[
\begin{aligned}
x'_s &= f_{c_o}(x) = \begin{cases} 
1 & s \text{ lies inside the boundary } c_o \\
n_s & s \text{ lies outside the boundary } c_o
\end{cases}.
\end{aligned}
\]

Notice that the flipped version, \( x'_s \), no longer has a boundary along \( c_o \). Furthermore, we can define the set of all flipped configurations as

\[
\Omega'_{c_o} \triangleq \{x' \in \Omega : x' = f_{c_o}(x) \text{ for some } x \in \Omega_{c_o}\}.
\]

\(^5\)Of course, diagonal neighbors to the curve can be \(-1\), and so can pixels that do not neighbor the curve.
Figure 14.5: In 2-D, the Ising model exhibits a unique critical temperature behavior in which the correlations between pixels in the model can extend to infinite distances. a) This figure illustrates the approach first used by Peierls and latter adapted by Kinderman and Snell to prove the existence of the critical temperature. b) This plot shows the exact value of the mean field strength of an infinite lattice with a boundary condition of $X_B = 1$. The analytic result comes from Onsager’s closed-form analysis of the Ising model as $N \to \infty$.

So the elements of $\Omega'_c$ are just the flipped versions of the elements of $\Omega_c$. Now, notice that with this definition, each element of $\Omega'_c$ corresponds to a unique element of $\Omega_c$; so the elements in the two sets are in a one-to-one correspondence, and the sets are assured to have the same size, i.e. $|\Omega_c| = |\Omega'_c|$.

Finally, we observe that for each element $x \in \Omega$, the total boundary length in the flipped version is shortened by $L(c(x))$. So the total boundary length in $x$ and $f(x)$ must be related as

$$L(f(x)) = L(x) - L(c_o),$$  \hspace{1cm} (14.12)

where $L(c_o)$ denotes the length of the curve $c_o$.

With these facts in mind, we can now upper bound the probability of the
event that a particular boundary $c_o$ occurs along with $X_0 = -1$, 

$$P_N\{c(X) = c_o, X_0 = -1 | B = 1 \} = \frac{1}{z} \sum_{x \in \Omega_{c_o}} \exp \{-\beta L(x)\}$$

$$= \sum_{x \in \Omega_{c_o}} \exp \{-\beta L(x)\}$$

$$\leq \sum_{x \in \Omega_{c_o}} \exp \{-\beta L(x)\}$$

$$= \sum_{x \in \Omega_{c_o}} \exp \{-\beta L(f(x))\}$$

$$= \sum_{x \in \Omega_{c_o}} \exp \{-\beta L(x)\}$$

$$= e^{-\beta L(c_o)},$$

where line three uses the fact that $\Omega_{c_o}' \subset \Omega$, line four uses the fact that elements of $\Omega_{c_o}$ and $\Omega_{c_o}'$ are in one-to-one correspondence, and line five uses equation (14.12). So this results in the bound that

$$P_N\{c(X) = c_o, X_0 = -1 | B = 1 \} \leq e^{-\beta L(c_o)}, \quad (14.13)$$

Now to bound $P_N\{X_0 = -1 | B = 1 \}$, we must sum the upper bound of equation (14.13) over all possible boundaries $c_o$. To do this, we define the set $C_L$ to contain all boundaries, $c_o$, of length $L$, such that the interior of $c$
includes the center pixel $X_0$. Then using this definition, we have that

$$P_N\{X_0 = -1|B = 1\} = \sum_{L=4}^{\infty} \sum_{c_o \in C_L} P\{c(x) = c_o, X_0 = -1|B = 1\}$$

$$\leq \sum_{L=4}^{\infty} \sum_{c_o \in C_L} e^{-\beta L}$$

$$= \sum_{L=4}^{\infty} |C_L| e^{-\beta L} .$$

So then the question becomes, how many such boundaries of length $L$ are there? Well, the boundaries must have even length, so $|C_L| = 0$ for $L$ odd. The boundary must also be within $L/2$ of the point $X_0$; so there can only be $L^2$ ways to select the starting point of the boundary, and once chosen, only 4 possibilities for the initial direction. For each subsequent step, there are only 3 possible choices of direction. Putting this together, we have that $|C_L| \leq 4L^23^L$ for $L$ even. Setting $L = 2n$, this leads to

$$P_N\{X_0 = -1|B = 1\} \leq \sum_{n=2}^{\infty} 4(2n)^23^{2n} e^{-\beta 2n}$$

$$= \sum_{n=2}^{\infty} 16n^2 (9e^{-2\beta})^n .$$

It is clear that for $\beta$ sufficiently large, we can ensure that

$$P_N\{X_0 = -1|B = 1\} \leq \sum_{n=2}^{\infty} 16n^2 (9e^{-2\beta})^n < 1/3 ;$$

so we have proved the desired result.

Amazingly, in 1944, Onsager was able to obtain closed form expressions for properties of the Ising model as $N \rightarrow \infty$ [49]. One representative result from this analysis is the following expression for the conditional expectation of a pixel in the lattice given that the boundary at infinity is $X_B = 1$.

$$\lim_{N \rightarrow \infty} E_N[X_0|B = 1] = \begin{cases} \left(1 - \frac{1}{\sinh(\beta)}\right)^{1/8} & \text{if } \beta > \beta_c \\ 0 & \text{if } \beta < \beta_c \end{cases} \quad (14.14)$$
This expression, plotted in Figure 14.5b, clearly illustrates the critical temperature behavior of the Ising model. For values of $\beta$ below a critical value of $\beta_c \approx 0.88$, the expected value of a pixel in the field is zero. This is what you would expect if the effect of the boundary were to become negligible as it tends towards infinity. However, for $\beta > \beta_c$ (i.e., $T < T_c$) where $\beta_c \approx 0.88$, this symmetry is broken, and the expectation moves towards 1. In fact, for large values of $\beta$, the expectation approaches 1, which means that $X_0 = 1$ with high probability.

Intuitively, this happens because when $\beta$ is above the critical value, the random field’s distribution becomes bi-modal, tending to be either nearly all values of 1 or nearly all values of $-1$. Clearly, without the boundary condition, the probability of $X$ and $-X$ (i.e., the random field with the sign of each pixel reversed) must be the same due to the symmetry in the density of equation 14.6. However, the assumption of a boundary of $B = 1$ breaks this symmetry in the model, and favors the mode associated with a majority of 1’s in the lattice. In this case, values of $-1$ only occur in primarily isolated locations.

The Ising model can be generalized to a wider range of similar models, but typically in these more general cases, it is no longer possible to compute closed form solutions to properties of the model. A simple and very useful generalization of the Ising model is to an $M$-level discrete MRF with typically an 8-point neighborhood. So for a 2-D lattice, let $C_1$ denote the set of horizontal and vertical cliques and let $C_2$ denote the set of diagonal cliques. Then we may define the following two statistics

$$t_1(x) \triangleq \sum_{\{s,r\} \in C_1} \delta(x_r \neq x_s) \quad (14.15)$$

$$t_2(x) \triangleq \sum_{\{s,r\} \in C_2} \delta(x_r \neq x_s) \quad (14.16)$$

where $t_1(x)$ counts the number of vertically and horizontally adjacent pixels with different values, $t_2(x)$ counts the number of diagonally adjacent pixels with different values, and the pixels $x_s$ are assumed to take values in the set $\{0, \cdots, M-1\}$. From this, we can express the distribution of an $M$-level MRF as

$$p(x) = \frac{1}{z} \exp \{-\beta_1 t_1(x) + \beta_2 t_2(x)\} \quad (14.17)$$
We can also compute the conditional distribution of a pixel, $X_s$, given its neighbors, $X_{\partial s}$. In order to compactly express this, we can first define the functions $v_1$ and $v_2$ which count the number of horizontal/vertical and diagonal neighbors with a different value from $X_s$. These functions are given by

$$v_1(x_s, x_{\partial s}) \triangleq \sum_{r \in [\partial s]_1} \delta(x_s \neq x_r)$$

$$v_2(x_s, x_{\partial s}) \triangleq \sum_{r \in [\partial s]_2} \delta(x_s \neq x_r),$$

where $[\partial s]_1$ is the set of horizontally and vertically adjacent neighbors to $s$, and $[\partial s]_2$ is the set of diagonally adjacent neighbors to $s$. Using this notation, we can then express the conditional distribution of $X_s$ given its neighbors as

$$p(x_s | x_r \neq s) = \frac{\exp \left\{-\beta_1 v_1(x_s, x_{\partial s}) - \beta_2 v_2(x_s, x_{\partial s})\right\}}{\sum_{m=0}^{M-1} \exp \left\{-\beta_1 v_1(m, x_{\partial s}) - \beta_2 v_2(m, x_{\partial s})\right\}}.$$

In this form, it is clear that when $\beta_1, \beta_2 > 0$, the pixel $X_s$ is more likely to take on a value that is shared by a number of its neighbors.

### 14.4 2-D Markov Chains and MRFs

In Section 14.2, we showed that the 1-D Markov chain and 1-D MRF represent equivalent models. Therefore, as a practical matter, a Markov chain model is almost always preferred in 1-D since, in many ways, it is more tractable.

So this raises the question, are 2-D Markov chains and 2-D MRFs equivalent? The answer to this is “no”. We will show that MRFs are more general in two or more dimensions, or when the lattice forms a general graph rather than a 1-D sequence or tree. In fact the relationship of Markov chains to MRFs in 1 and 2 dimensions, is largely analogous the relationship between AR models and Gaussian MRFs as described in Chapter 3.

We will start by first defining $P^{th}$ order Markov chains and Markov random fields in 2-D. To do this, we will need to re-introduce the 2-D causal ordering of points in a plane, as first described in Section 3.4. Figure 3.3b illustrates the 2-D window of causal neighbors of a point using raster ordering. More
Figure 14.6: Figure illustrating how a 2-D Markov chain of order \( P \) is still an MRF, but with an order of \( 2P \). (a) Illustration of the lattice points that form cliques in the Gibbs distribution of a 2-D Markov chain. (b) The associated MRF neighborhood is formed by the autocorrelation of the clique structure and results in an asymmetric neighborhood system.

formally, the causal neighbors of the point \( s \) are the points \( s - r \), where \( r \in W_P \), and \( W_P \) is a \( P^{th} \) order window as defined in equation (3.12).

Using the same approach as in Section 14.2, we first define a 2-D Markov chain of order \( P \).

**Definition 25.** \( P^{th} \) order 2-D Markov Chain

Let \( S \) be a 2-D rectangular lattice. Then \( \{X_s\}_{s \in S} \) is said to be a \( P^{th} \) order 2-D Markov chain if for all \( s \), the following holds,

\[
P\{X_s = x_s | X_r = x_r \text{ for } r < s\} = P\{X_s = x_s | X_{s-W_P} \}
\]

where \( W_P \) is defined as in equation (3.12) of Section 3.4.

In order to simplify notation, let \( \{s - W_P\} \) denote the set of previous pixel locations in the Markov chain. Figure 14.6a illustrates that the set \( \{s - W_P\} \) is an asymmetric causal window which only includes pixels that occurred before \( s \) in raster order. Now if \( X_s \) is a 2-D Markov chain of order \( P \), we can write the joint distribution of \( X \) as:

\[
p(x) = \prod_{s \in S} P\{X_s = x_s | X_{s-W_P} = x_{s-W_P}\}.
\]

\(^6\)For notational simplicity, we ignore the initialization terms of the Markov chain that occur along the boundaries of a finite MRF. In fact, the expression still holds as long as the conditional probability is interpreted to be simply the probability when the conditioning variables are unavailable.
If we also assume the technical condition that \( p(x) \) is strictly positive, then we know we can write this as a Gibbs distribution with the form

\[
p(x) = \frac{1}{z} \exp \left\{ - \sum_{s \in S} V_s(x_s, x_{s-W_P}) \right\},
\]

(14.18)

where the potential functions are defined by

\[
V_s(x_s, x_{s-W_P}) = -\log P\{X_s = x_s|X_{s-W_P} = x_{s-W_P}\}.
\]

From the form of equation (14.18), we can see that the 2-D Markov chain has a Gibbs distribution.

By the Hammersley-Clifford Theorem, we know that since \( X_s \) has a Gibbs distribution, it must be an MRF with associated cliques of \( c_s = \{s, s - W_P\} \), and a complete set of cliques given by \( C = \{c_s : s \in S\} \). From this we may determine the neighborhood system of the MRF. In particular, we know from equation (14.4) that \( \partial s \) must contain any pixel, \( r \), that shares a clique with \( s \). With a little thought, it becomes clear that the associated neighborhood system for this set of cliques can be determined by performing the 2-D autocorrelation of the cliques, \( c_s \), with themselves. The result is shown in Figure 14.6b. In fact, the relationship between these cliques and the neighborhood of the MRF is precisely the same relationship as discussed in Section 4.5 for the Gaussian MRF with pairwise cliques.

From this, we can define a 2-D Markov random field of order \( P \). The natural definition is that each pixel is dependent on a \((2P + 1) \times (2P + 1)\) non-causal neighborhood of surrounding pixels. So with this definition, and our previous results, we can see that a 2-D Markov chain of order \( P \) must be a 2-D MRF of order \( 2P \).

**Definition 26.** \( P \)th order 2-D Markov Random Field

Let \( S \) be a 2-D rectangular lattice with neighborhood system

\[
\partial(s_1, s_2) = \{(r_1, r_2) \in S : 0 < \max\{|r_1 - s_1|, |r_2 - s_2|\} \leq P\}.
\]

Then \( \{X_s\}_{s \in S} \) is said to be a \( P \)th order 2-D Markov random field if for all \( s \), the following holds,

\[
P\{X_s = x_s|X_r = x_r \text{ for } r \neq s\} = P\{X_s = x_s|X_{\partial s}\}.
\]
Since in 2-D a Markov chain is an MRF, one might again wonder if an MRF can be represented as a Markov chain? In fact, this is generally not the case. So as an example, consider the Ising model of equation (14.7). In this case, the conditional distribution of a pixel, \( X_s \), given the past pixels, \( X_r \) for \( r < s \), and be computed as

\[
P\{X_s = x_s | X_r = x_r, r < s\} = \frac{P\{X_s = x_s, X_r = x_r, r < s\}}{\sum_{m=0}^{M-1} P\{X_s = m, X_r = x_r, r < s\}}.
\]

However, in order to compute the joint density of the numerator, we need to perform a sum over all pixels \( x_k \) for \( k > s \),

\[
P\{X_s = x_s, X_r = x_r, r < s\} = \sum_{x_k \text{ for } k > s} \frac{1}{z(\beta)} \exp\left\{-\beta \sum_{\{i,j\} \in C} \delta(x_i \neq x_j)\right\},
\]

where again \( k = (k_1, k_2) \) denotes a vector index. Some manipulation of these terms then results in a conditional distribution with the form

\[
P\{X_s = x_s | X_r = x_r, r < s\} = P\{X_s = x_s | X_r = x_r, r \in B_s\},
\]

where \( B_s \) is a set of boundary pixels that precede the pixel \( s \) and lie along two rows of the image.

Figure 14.7: This figure illustrates the causal neighborhood of a pixel \( s \) which makes \( X_s \) conditionally independent of the past for an Ising model with a 4-point neighborhood. The symbol \( \otimes \) denotes the location of the pixel \( s \), and the set of points labeled \( B_s \) indicate the positions necessary to make \( X_s \) conditionally independent of the past.

Figure ?? illustrates the structure of this set, \( B_s \), relative to the pixel \( s \). Importantly, the set \( B_s \) extends indefinitely to the left and right. This means the the random field \( X_s \) is not a 2-D Markov chain for any fixed finite value of \( P \). Consequently, even with this simple example of the Ising model, we can see that MRFs are not generally 2-D Markov chains. In fact, this makes
Figure 14.8: Venn diagram illustrating the relationship between Markov chains (MC) and Markov Random Fields (MRF) in 1 and 2-D among the set of all discrete-state random processes with strictly positive distribution. Notice that in 1-D, Markov chain and MRF models are equivalent, but in 2-D they are not.

sense because if all MRFs were Markov chains, then we would expect that two pixels, $X_s$ and $X_r$, would become decorrelated when the lattice points $s$ and $r$ become sufficiently far apart. However, we know from the discussion of critical temperature in Section 14.3 that this does not happen when $\beta$ is greater than the critical value.

Figure 14.8 illustrates the relationships between Markov chains and MRFs graphically with a Venn diagram. The Venn diagram only treats discrete random processes with PMFs that are strictly positive, but the results typically apply to a broader class of continuous random processes with well-defined densities. Notice that in 1-D, Markov chains and MRFs parameterize the same family of models, so they are equally expressive. However, in 2-D, MRFs are more general than Markov chains. More specifically, 2-D Markov chains of order $P$ are always 2-D MRFs, but of order $2P$.

### 14.5 MRF Parameter Estimation

Perhaps the greatest practical deficiency of MRFs is the general difficulty of estimating parameters of MRF models. In fact, typical MRF models have a relatively small number of parameters in order to minimize the difficulties associated with parameter estimation.

The difficulty in MRF parameter estimation results from the fact that the
partition function is typically not a tractable function of the parameters. So for example, in the general case, the MRFs distribution can be written as a Gibbs distribution with the form

\[ p_\theta(x) = \frac{1}{z(\theta)} \exp \{ -u_\theta(x) \} , \]

where \( u_\theta(x) \) is the energy function parameterized by \( \theta \), and \( z(\theta) \) is the partition function. In this form, the ML estimate is given by

\[ \hat{\theta} = \arg \max_\theta \log p_\theta(x) \]

\[ = \arg \min_\theta \{ u_\theta(x) + \log z(\theta) \} . \]

Typically, the ML estimate is difficult to compute because the term \( z(\theta) \) does not have a tractable form. However, it is still possible to make a bit more progress in analyzing this expression. Assuming the the quantities are all differentiable and that the ML estimate does not lie on a constraint boundary, then a necessary condition is that the ML estimate, \( \hat{\theta} \), solve the following equation,

\[ \nabla u_\hat{\theta}(x) = -\frac{\nabla z(\hat{\theta})}{z(\hat{\theta})} , \]

where the gradient is with respect to \( \theta \). Now by using the definition of the partition function, and applying some simple calculus, it is easily shown that

\[ -\frac{\nabla z(\hat{\theta})}{z(\hat{\theta})} = \sum_{x \in \Omega^N} \nabla u_\hat{\theta}(x) \frac{1}{z(\hat{\theta})} \exp \{ -u_\hat{\theta}(x) \} \]

\[ = \mathbb{E}_\theta [ \nabla u_\hat{\theta}(X) ] . \]

So the ML estimate of the parameter, \( \hat{\theta} \), is given by the solution to the equation

\[ \nabla u_\hat{\theta}(x) = \mathbb{E}_\theta [ \nabla u_\hat{\theta}(X) ] . \] (14.19)

It is often the case that the Gibbs distribution is parameterized as an exponential family so that it can be written as

\[ u_\theta(x) = T(x)\theta \]

where \( T(x) \) is a \( 1 \times k \) row vector and \( \theta \) is a \( k \times 1 \) column vector. For instance, this is the case in the example of equation (14.17) where \( T(x) = [t_1(x), t_2(x)] \).
and \( \theta = [\beta_1, \beta_2]^t \). In this case, \( \nabla u_\theta(x) = T(x) \), and the relationship of equation (14.19) becomes

\[
T(x) = \mathbb{E}_\theta [T(X)] .
\] (14.20)

Equation (14.20) has the intuitively appealing interpretation that, to compute the ML estimate, one must adjust the components of the parameter \( \theta \) until the expected values of the statistics, (i.e., \( \mathbb{E}_\theta [T(X)] \)) match the observed values of the statistics, (i.e., \( T(x) \)).

Of course, the difficulty in solving equation (14.20) is computing the required expectation. One approach to solving this problem numerically is to approximately compute the expectation using the simulation techniques of that will be introduced in Chapter 15. Methods such as the Metropolis, Hastings-Metropolis, and Gibbs sampler provide techniques to generate sequences of samples from the distribution of \( X \) using parameter \( \theta \). So for example, simulation methods can be used to generate \( M \) samples from the Gibbs distribution with the parameter \( \theta \). Then the following error can be computed as

\[
\Delta T \leftarrow T(x) - \frac{1}{M} \sum_{m=1}^{M} T(X_\theta^{(m)}) ,
\]

where \( \{X_\theta^{(m)}\}_{m=1}^{M} \) are the \( M \) samples of \( X \) generated from the distribution with parameter \( \theta \). Using this error, the parameters of \( \theta \) can be adjusted by decreasing its components in proportion to the error

\[
\theta \leftarrow \theta - \alpha \Delta T ,
\]

where \( \alpha \) is a user selected step size. In any case, this approach to ML estimation can be very computationally expensive since it requires the use of stochastic sampling techniques, and its convergence can be fragile.

A much more commonly used approach to ML estimation for MRF parameters is the so called maximum pseudo-likelihood (MPL) estimator [7, 8]. The MPL estimator works by replacing the true likelihood with the so-called pseudo likelihood that does not include the dreaded partition function. Since \( X \) is assumed to be an MRF, we know that it is possible to write the conditional distributions of pixels given all other pixels as \( p_\theta(x_s|x_{\partial s}) \), where \( \partial s \) is the set of neighbors to the pixel \( s \). So the pseudo-likelihood function is
defined as
\[ PL(\theta) = \prod_{s \in S} p_\theta(x_s|x_{\partial s}) . \]

Of course, the pseudo-likelihood is not the actual likelihood because the dependencies in the MRF are loopy. Nonetheless, we can use the pseudo-likelihood as a criteria for optimization in parameter estimation. More specifically, the maximum pseudo-likelihood estimate of the parameter \( \theta \) is then given by
\[ \hat{\theta} = \arg \max_{\theta} \left\{ \sum_{s \in S} \log p_\theta(x_s|x_{\partial s}) \right\} . \]

So the maximum pseudo-likelihood estimate is based on the strategy of simply ignoring the loopy nature of the MRF dependencies and computing the likelihood as if the probabilities formed a Markov chain.

However, one glitch in the approach is that in applications the parameter \( \theta \) typically is used to parameterize the Gibbs distribution rather than the conditional probabilities themselves\(^7\). So therefore, using the MPL approach requires that the conditional distributions must be expressed in terms of the associated Gibbs distribution using Bayes’ rule. This results in a complex and typically nonlinear relationship between the parameters, \( \theta \), and the psuedo-likelihood expression to be optimized. So consider a Gibbs distribution with the form
\[ p_\theta(x) = \frac{1}{z} \exp \left\{ - \sum_{c \in C} V_c(x_c|\theta) \right\} . \]

Once again, we can define \( C_s \) to be the set of all cliques that contain the pixel \( x_s \). With this notation, it is easily shown that
\[ p_\theta(x_s|x_{\partial s}) = \frac{\exp \left\{ - \sum_{c \in C_s} V_c(x_c|\theta) \right\}}{\sum_{x_s \in \Omega} \exp \left\{ - \sum_{c \in C_s} V_c(x_c|\theta) \right\}} . \]

So using this expression, the functional form of the maximum pseudo-likelihood estimate is given by
\[ \hat{\theta} = \arg \max_{\theta} \left\{ \sum_{s \in S} \log \left( \frac{\exp \left\{ - \sum_{c \in C_s} V_c(x_c|\theta) \right\}}{\sum_{x_s \in \Omega} \exp \left\{ - \sum_{c \in C_s} V_c(x_c|\theta) \right\}} \right) \right\} . \quad (14.21) \]

\(^7\) It is generally necessary to parameterize the Gibbs distribution rather than the conditional probabilities because the Hammersley-Clifford Theorem does not say how to construct the full distribution from the conditional distributions. Also, a given set of conditional distributions are not likely to consistently correspond to any overall joint distribution.
While this expression is not very pretty, it can be numerically optimized. In particular, it is tractable because the sum in the denominator of each term is only over individual pixels, $x_s$, rather than the entire MRF. However in practice, this expression tends to be a highly nonlinear function of the parameter $\theta$, so numerical optimization can be tricky. Nonetheless, an important advantage of the maximum pseudo-likelihood estimate is that it has been shown to be consistent \cite{29}, so it is typically better than completely ad-hoc parameter estimates, which often are not.
1. In the following problem, we show that for every neighborhood system, there is a unique set of cliques, and for every set of cliques there is a unique neighborhood system.

a) Show that for any neighborhood system, $\partial s$, there is a unique largest set of cliques, $\mathcal{C}$. We say that the set of cliques $\mathcal{C}$ is larger than the set of cliques $\mathcal{C}'$ if $\mathcal{C}' \subset \mathcal{C}$.

b) Show that for any set of cliques, $\mathcal{C}$, there is a unique smallest neighborhood system. Here we say that the neighborhood system $\partial s$ is smaller than the neighborhood system $\partial s'$, if for all $s \in S$, $\partial s \subset \partial s'$.

2. Let $X$ be a discrete valued random field on a lattice, and assume that its PMF has the form of a Gibbs distribution with neighborhood system $\partial s$ and associated cliques $\mathcal{C}$. Then show that $X$ is a MRF with neighborhood system $\partial s$.

3. Let $\{X_n\}_{n=0}^{N}$ be a $P^{th}$ order Markov chain. Furthermore, define the new random process, $\{Y_n\}_{n=0}^{N}$, given by $Y_n = (X_n, \cdots, X_{\max\{n-P,0\}})$. Then show that $Y_n$ is a Markov chain of order $P = 1$.

4. Prove that in 1-D a $P^{th}$ order Markov chain is also a $P^{th}$ order MRF.

5. Prove that in 1-D a $P^{th}$ order MRF is also a $P^{th}$ order Markov chain.

6. In this problem, we will derive the general form of the discrete distribution known as the Boltzmann or Gibbs distribution of Appendix B.

a) Derive the result of equation (B.2).

b) Derive the result of equation (B.3).

c) Derive the results of equations (B.5), (B.6), and (B.7).

d) For a system in thermodynamic equilibrium, is it possible for the temperature, $T$, to be negative? Use the Ising model to propose an example of a physical system with negative temperature.

7. Using the Ising distribution specified by equation (14.7), do the following:
a) Show that in 2-D the conditional distribution of a pixel given its neighbors has the form of equation 14.11.

b) Plot the conditional probability \( p(x_s = 1|x_i \neq s) \) as a function of \( v(1, x_{\partial s}) \) with \( \beta = -1, 0, 0.5, \) and 1.

c) What would a “typical” sample of the \( X \) look like when \( \beta = -2? \)

8. Answer the following questions using results from Example 14.1 and equation (14.14).

a) From the example, calculate a value of \( \beta \) which guarantees that \( P_N \{X_0 = -1|X_B = 1\} < 1/3 \) for all \( N > 3 \).

b) Find a value of \( \beta \) which ensures that the \( \lim_{N \to \infty} \mathbb{E}_N \{ X_s | X_B = 1 \} \geq 0.9 \) for all lattice points \( s \).

c) Does the proof of Example 14.1 hold for a 1-D Ising model? If so, prove the result. If not, explain why?

9. Let \( X \) be a binary valued 2-D random field with \( N \times N \) points. Assume that for \( 0 < i, j \leq N - 1 \)

\[
P\{X_{(0,j)} = 0\} = P\{X_{(i,0)} = 0\} = \frac{1}{2}
\]

and for

\[
P\{X_{(i,j)} = x_{(i,j)} | X_r = x_r \ r < (i,j)\} = g(x_{(i,j)}|x_{(i-1,j)}, x_{(i,j-1)})
= \frac{1}{3}\delta(x_{(i,j)}, x_{(i-1,j)}) + \frac{1}{3}\delta(x_{(i,j)}, x_{(i,j-1)}) + \frac{1}{6}
\]

a) Compute the complete density function for \( X \).

b) Show that \( X \) is an MRF. Give the cliques and the neighborhood for \( X \).
Chapter 15

Stochastic Simulation

So far, we have reviewed many non-causal image and signal models, and we have shown how they can be of great value as prior distributions in inverse problems such as image restoration and deconvolution. However, we have not yet given any approach for generating samples from these distributions.

The objective of this chapter is to present a general class of techniques for stochastic sampling from complex and high-dimensional distributions. These techniques, which we broadly refer to as stochastic simulation, are based on the use of Monte Carlo methods for pseudo-random sampling of distributions. While we present stochastic simulation as a method for generating samples of random objects, the methods can be used to simulate a broad range of practical stochastic systems in applications ranging from signal processing, to communications, to chemistry and material science. In addition, these methods also serve as a basic tool used in a wide range of techniques including stochastic optimization, stochastic integration, and stochastic EM parameter estimation.

15.1 Motivation for Simulation Methods

In order to understand the motivation for stochastic simulation methods, consider a situation in which one would like to use a computer to generate a sample of a random variable $X$ with a CDF of $F(x) = P\{X \leq x\}$. The classic approach to this problem is to first generate a pseudo-random number which is uniformly distributed on the interval $[0, 1]$,

$$U \leftarrow \text{Rand}([0, 1])$$
where Rand([0, 1]) denotes a process for generating such a random variable. Next, one calculates the inverse CDF defined by

\[ F^{-1}(u) \triangleq \inf \{ x \mid F(x) \geq u \} . \]

Then it can be shown that the desired random variable can be generated by applying the inverse CDF to the uniform random variable.

\[ X \leftarrow F^{-1}(U) \quad (15.1) \]

In fact, it can be proven that this inverse transform sampling method works for any random variable. (See problem 2 of Chapter 2)

With this result, one might think it would be easy enough to generate a random variable with any desired distribution. However, this is not always the case. For some distributions, like for example exponential or Gaussian random variables,\(^1\) it is possible to calculate the inverse CDF in closed form, so that the inverse transform method of equation (15.1) can be directly implemented. However, for a general random variable, the inverse CDF typically does not have any simple form; so that direct implementation is not tractable.

For example, consider the random variable, \( X \), with a density function given by

\[ p(x) = \frac{1}{z} \exp \left\{ - \left( 4(x - 1)^2(x + 1)^2 + \frac{x}{2} \right) \right\} . \quad (15.2) \]

where \( z \) is a normalizing constant for the density. In fact, this density has the form of a Gibbs distribution, \( p(x) = \frac{1}{z} e^{-u(x)} \), where

\[ u(x) = 4(x - 1)^2(x + 1)^2 + \frac{x}{2} . \]

From Figure 15.1, we can see that both the energy function and its associated density are bimodal for this distribution, with modes peaking at approximately \( x = \pm 1 \). Not surprisingly, the CDF of this density does not have closed analytical form, so neither does its inverse. So how might one efficiently generate samples of this random variable on a computer?

In the following sections, we present a set of tools broadly termed as stochastic simulation methods, that can be used to generate samples from

---

\(^1\) It is interesting to note that the inverse CDF does not have closed form for the Gaussian distribution, but this problem can be circumvented by generating the distribution of pairs of independent Gaussian random variables since their vector norm is exponentially distributed.
The Metropolis Sampler

The Metropolis sampler is a very clever method that allows samples to be generated from any Boltzmann or equivalently Gibbs distribution \([46]\). Con-
Consider the discrete distribution on the set \( \Omega \) with a PMF of the form
\[
p(x) = \frac{1}{z} \exp \{-u(x)\},
\]  
(15.3)
where \( u(x) \) is an energy function and \( z \) is the associated partition function. So any strictly positive PMF can be put in this form since it is always possible to take the energy function as
\[
u(x) = -\log p(x) + \text{const}.
\]
In fact, the random quantity \( X \) may be a random variable, or it may be a higher dimensional random object such as a vector or random field.

The objective of a simulation algorithm is to generate a sequence of random objects, denoted by \( X^{(k)} \), whose distribution converges to the desired Gibbs distribution, \( p(x) \). The strategy of the Metropolis algorithm is to use the current sample, \( X^{(k)} \), to generate a “proposal” for a new sample denoted by, \( W \). The proposal, \( W \), is generated randomly, but its distribution is dependent on the current sample, \( X^{(k)} \). More specifically, the proposed sample \( W \) is chosen from some conditional distribution \( q(w|X^{(k)}) \). However, this proposal density, \( q(w|x^{(k)}) \), must obey the symmetry property that for all \( w, x \in \Omega \),
\[
q(w|x) = q(x|w).
\]  
(15.4)
While the condition of (15.4) is quite restrictive, there are reasonable methods for generating proposals that fulfill the condition. For example, we can set the proposal to be \( W \leftarrow X^{(k)} + Z \), where \( Z \) is a random variable with a symmetric PDF that is independently generated at each new time \( k \). In this case, it is easy to show that the condition applies.

In practice, the proposal \( W \) can be generated with the inverse transform sampling method of equation (15.1). In pseudocode, this can be expressed as
\[
U \leftarrow \text{Rand}([0, 1]) \quad \quad W \leftarrow Q^{-1}(U|X^{(k)}) ,
\]
where \( Q^{-1}(\cdot|X^{(k)}) \) is the inverse CDF corresponding to the density \( q(w|x^{(k)}) \). The proposal density is typically chosen to be relatively simple, so that generating samples from this distribution is easy.

\(^2\)The theory we develop here assumes discrete random variables, but the methods equally apply to the sampling of continuously valued random variables. In fact, the extension to continuous random variable is largely intuitive since the continuous random variables can be quantized to form discrete ones.
The clever trick of the Metropolis algorithm is then to accept or reject these proposals in a very particular way. To do this, we first compute the change in the energy given by \( \Delta E \leftarrow u(W) - u(X^{(k)}) \). If the change in energy is negative, \( \Delta E < 0 \), then we immediately accept the proposal, so that
\[
X^{(k+1)} \leftarrow W \quad \text{if } \Delta E < 0 .
\]
However, if the change in energy is positive, \( \Delta E \geq 0 \), then we accept the proposal with a probability of \( e^{-\Delta E} \) and reject it with probability \( 1 - e^{-\Delta E} \).

In practice, this is done with the following two steps.
\[
U \leftarrow \text{Rand}([0, 1]) \\
X^{(k+1)} \leftarrow W \quad \text{if } \Delta E \geq 0 \text{ and } U < e^{-\Delta E} \\
X^{(k+1)} \leftarrow X^{(k)} \quad \text{if } \Delta E \geq 0 \text{ and } U \geq e^{-\Delta E} .
\]

We can simplify this acceptance decision by introducing a function that expresses the acceptance probability for each proposal.
\[
\alpha(w|x^{(k)}) = \min \left\{ 1, e^{-[u(w)-u(x^{(k)})]} \right\} \tag{15.5}
\]

Using this function, the Metropolis sampling is performed in the following steps. First, the proposal, \( W \), is generated according to the sampling distribution.
\[
U \leftarrow \text{Rand}([0, 1]) \\
W \leftarrow Q^{-1}(U|X^{(k)})
\]

Then the sample is either accepted or rejected according to the acceptance probability \( \alpha(W|X^{(k)}) \).
\[
U \leftarrow \text{Rand}([0, 1]) \\
X^{(k+1)} \leftarrow W \quad \text{if } U < \alpha(W|X^{(k)}) \\
X^{(k+1)} \leftarrow X^{(k)} \quad \text{if } U \geq \alpha(W|X^{(k)}) .
\]

Figure [15.2] lays out the Metropolis algorithm in pseudocode format. The intuition behind the algorithm is that decreases in energy are always accepted but the probability of accepting a proposal with an increasing energy decreases exponentially as the energy increases.
Figure 15.2: A pseudocode specification of the Metropolis algorithm for stochastic simulation from Gibbs distributions.

While this may all seem reasonable, how do we know that the Metropolis algorithm converges to the desired distribution of equation (15.3)? The proof of convergence depends on a clever use of ergodic theory for Markov chains. First notice that the sequence \( \{X(k)\}_{k=1}^N \) is a Markov chain. This is because each new sample, \( X(k+1) \), is only dependent on the previous sample, \( X(k) \), and other independently generated random variables.

Next, we can show (perhaps surprisingly) that this Markov chain is reversible as defined in Section 13.4. To see why this is true, consider the transition probabilities for the Markov chain denoted by \( p(x^{(k+1)}|x^{(k)}) \). With a little thought it is clear that the transition probability is given by

\[
p(x^{(k+1)}|x^{(k)}) = \begin{cases} 
q(x^{(k+1)}|x^{(k)}) \alpha(x^{(k+1)}|x^{(k)}) & \text{if } x^{(k+1)} \neq x^{(k)} \\
1 - \sum_{w \neq x^{(k)}} q(w|x^{(k)}) \alpha(w|x^{(k)}) & \text{if } x^{(k+1)} = x^{(k)}
\end{cases} \tag{15.6}
\]

The justification for this expression is that when \( x^{(k+1)} \neq x^{(k)} \), then the probability of the transition is the product of the probabilities that you will generate the proposal multiplied by the probability that you will accept the proposal. Of course, since the transition probabilities must sum to one, the probability that \( x^{(k+1)} = x^{(k)} \) is just one minus the sum of the alternative cases.

In order to show that these transition probabilities are reversible, we must show that for all state combinations, \( (x, x') \), the detailed balance equations
of (13.29) hold. For our problem, these are given by
\[
p(x'|x) \frac{1}{z} e^{-u(x)} = p(x|x') \frac{1}{z} e^{-u(x')},
\]
where \( p(x) = \frac{1}{z} e^{-u(x)} \) is the stationary distribution of the ergodic Markov chain.

To show this, we will consider three cases: case 1 in which \( u(x') \leq u(x) \) and \( x' \neq x \); case 2 in which \( u(x') > u(x) \) and \( x' \neq x \); and case 3 in which \( x' = x \). For case 1, we have have that
\[
p(x'|x) = q(x'|x) \alpha(x'|x)
= q(x'|x)
= q(x|x')
= q(x|x') e^{-[u(x)-u(x')] [u(x)-u(x')]}
= q(x|x') \alpha(x|x') e^{[u(x)-u(x')]}
= p(x|x') e^{[u(x)-u(x')]}. \tag{15.7}
\]
Rearranging terms and normalizing by the partition function \( z \) then provides the desired result.
\[
p(x'|x) \frac{1}{z} e^{-u(x)} = p(x|x') \frac{1}{z} e^{-u(x')} \tag{15.8}
\]
Now there are two additional cases to consider. Case 2 results from swapping the roles of \( x \) and \( x' \). However, the symmetry of equation (15.8) then immediately yields the desired result. For case 3, substituting in \( x = x' \) results in the two sides of equation (15.8) being identical; so the required equality holds trivially.

The detailed balance equations of (15.8), then imply the full balance equations hold by simply summing over the variable \( x \).
\[
\sum_{x \in \Omega} p(x'|x) \frac{1}{z} e^{-u(x)} = \frac{1}{z} e^{-u(x')}. \tag{15.9}
\]
So we can see that the Gibbs distribution \( p(x) = \frac{1}{z} e^{-u(x)} \) is in fact the stationary distribution of the Markov chain generated by the Metropolis algorithm.

By Theorem 13.4.1 of Section 13.4, we now know that if the proposal distribution \( q(x^{(k+1)}|x^{(k)}) \) is chosen so that all states can be accessed (i.e.,
Hastings-Metropolis Algorithm:

Initialize $X^{(0)}$

For $k = 0$ to $K - 1$

/*Generate proposal*/

$W \sim q(w|X^{(k)})$ /*Use proposal distribution to generate $W$*/

/*Accept or reject proposal*/

$\alpha \leftarrow \min\left\{1, \frac{q(X^{(k)}|W)}{q(W|X^{(k)})} \exp\{-[u(W) - u(X^{(k)})]\}\right\}$ /*Acceptance prob.*/

$X^{(k+1)} \leftarrow \begin{cases} W \text{ with probability } \alpha \\ X^{(k)} \text{ with probability } 1 - \alpha \end{cases}$

Figure 15.3: A pseudocode specification of the Hastings-Metropolis algorithm for stochastic simulation from Gibbs distributions. The Hastings-Metropolis algorithm improves on the more basic Metropolis algorithm by allowing for an asymmetric proposal distribution.

the Markov chain is irreducible), the probability of remaining in the same state is greater than zero (i.e., the Markov chain is aperiodic), and if the number of states are finite, then the Metropolis algorithm must generate an ergodic Markov chain that converges to the desired Gibbs distribution.

In this case, we have that

$$\lim_{k \to \infty} P\{X^{(k)} = x\} = \frac{1}{z} \exp\{-u(x)\}.$$

In fact, the Metropolis algorithm is commonly used even when these technical conditions do not hold. For example, it is commonly used to generate samples from high-dimensional continuous distributions with densities, such as non-Gaussian MRF models like those discussed in Chapter 6. In fact, much stronger convergence results exist for stochastic simulation methods and ergodic Markov chains, so our convergence result is simply provided to illustrate the basic concept.

15.3 The Hastings-Metropolis Sampler

While the Metropolis algorithm is a very powerful method, it can be slow to converge to the desired distribution. This is a problem for two important
reasons. First, it means that one might need to generate many samples before the ergodic distribution is reached. This time need to run the Markov chain until it approximately converges to its ergodic distribution is sometimes referred to as the **burn-in time**. But a second equally important consequence of slow convergence is that it requires more samples to be generated before the random variables $X^k$ become approximately independent. Naive users sometime assume that subsequent samples of the Markov chain are independent, but this is not true. So if the object is to produce independent stochastic samples from the Markov chain, as is often the case, then a slowly converging Markov chain will require more samples between each approximately independent sample.

A crucial limitation in the Metropolis algorithm is the requirement that the proposal distribution be symmetric. That is

$$q(x^{(k+1)}|x^{(k)}) = q(x^{(k)}|x^{(k+1)})$$

Intuitively, it is best if the proposal distribution closely approximates the desired ergodic distribution. However, this restriction to a symmetric distribution is severe because it is often the case that we might want to “bias” the proposals towards ones that more closely match the desired distribution and are therefore more likely to be accepted.\footnote{This strategy is also highly recommended when pursuing funding.} Intuitively, proposals that are rejected do not lead to any change in the state, so they are substantially wasted, and one would expect the associated Markov chain’s convergence to be slowed \cite{33}.

Fortunately, Hastings and Peskun discovered a generalization of the Metropolis sampler, generally referred to as the **Hastings-Metropolis sampler**, that allows for the use of proposal distributions that are not symmetric \cite{35,52}. So in this case, any valid distribution, $q(x^k|x^{k-1})$, can be chosen, but in order to correct for the bias due to this asymmetric proposal density, the probability of acceptance and rejection must be appropriately adjusted.

The acceptance probability for the Hastings-Metropolis sampler is given
by
\[
\alpha(w|x^{(k)}) = \min \left\{ 1, \frac{q(x^{(k)}|w)}{q(w|x^{(k)})} \exp \left\{ -[u(w) - u(x^{(k)})] \right\} \right\}
\]
\[
= \min \left\{ 1, \frac{q(x^{(k)}|w) p(w)}{q(w|x^{(k)}) p(x^{(k)})} \right\} .
\]
So the resulting transition probability is then given by
\[
p(x^{(k+1)}|x^{(k)}) = \begin{cases} 
q(x^{(k+1)}|x^{(k)}) \alpha(x^{(k+1)}|x^{(k)}) & \text{if } x^{(k+1)} \neq x^{(k)} \\
1 - \sum_{w \neq x^{(k)}} q(w|x^{(k)}) \alpha(w|x^{(k)}) & \text{if } x^{(k+1)} = x^{(k)}
\end{cases}.
\]
(15.10)

As with the basic Metropolis algorithm, our approach is to prove that the Markov chain resulting from the Hastings-Metropolis algorithm is reversible. To prove this, we must again show that the Gibbs distribution solves the detailed balance equation. Our proof is broken into three cases:

Case 1: \( x' \neq x \) and \( \frac{q(x'|x)}{q(x|x')} \exp \left\{ -[u(x') - u(x)] \right\} \geq 1; \]

Case 2: \( x' \neq x \) and \( \frac{q(x'|x)}{q(x|x')} \exp \left\{ -[u(x') - u(x)] \right\} \leq 1; \]

Case 3: \( x' = x \).

We again start with case 1 for which we have that
\[
p(x'|x) = q(x'|x) \alpha(x'|x)
\]
\[
= q(x'|x)
\]
\[
= q(x'|x) \frac{q(x'|x)}{q(x|x')} \frac{q(x|x')}{q(x'|x)} \exp \left\{ -[u(x) - u(x')] \right\} \exp \left\{ -[u(x') - u(x)] \right\}
\]
\[
= q(x'|x) \alpha(x'|x) \exp \left\{ -[u(x') - u(x)] \right\} = p(x'|x) \exp \left\{ -[u(x') - u(x)] \right\} ,
\]
(15.11)
From this we again get the detailed balance equation.
\begin{equation}
    p(x'|x) \frac{1}{z} e^{-u(x)} = p(x|x') \frac{1}{z} e^{-u(x')}
\end{equation}
(15.12)

Once again, case 2 results directly from the symmetry of the of equation (15.12), and case 3 is immediate by substituting $x' = x$ into equation (15.12).

Figure 15.3 presents a pseudocode version of the Hastings-Metropolis algorithm. The method is basically the same as for the Metropolis algorithm, except that the acceptance probability incorporates the ratio $\frac{q(X^{(k)}|W)}{q(W|X^{(k)})}$.

Another special case arrises when the proposal distribution is exactly the true Gibbs distribution. In this case, we have that $q(w|x) = p(x)$, where $p(x)$ is the desired distribution, and then
\[
    \alpha(w|x) = \min \left\{ 1, \frac{q(x|w)p(w)}{q(w|x)p(x)} \right\}
\]
\[
    = \min \left\{ 1, \frac{p(x)p(w)}{p(w)p(x)} \right\} = 1 .
\]

So in this case, the acceptance probability is always 1, and the algorithm converges to the Gibbs distribution in one step! (See problem 4)

**Example 15.1.** In this example, we use both the Metropolis and Hastings-Metropolis algorithms to generate samples from the distribution
\begin{equation}
    p(x) = \frac{1}{z} \exp \{-u(x)\} ,
\end{equation}
(15.13)
where
\[
    u(x) = 4(x - 1)^2(x + 1)^2 + \frac{x}{2} .
\]

The upper left hand plot of Figure 15.4 shows a plot of this density function. We will generate proposals, $W$, for the Metropolis algorithm as
\[
    W = X^{(k)} + Z ,
\]
where $Z$ is additive Gaussian noise with $Z \sim N(0, (1/2)^2)$. From this we know that the proposal distribution has the form
\[
    q(w|x^{(k)}) = \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left\{ -\frac{1}{2\sigma^2} \left( w - x^{(k)} \right)^2 \right\} ,
\]
Figure 15.4: Example illustrating the use of the Metropolis and Hastings-Metropolis algorithms for stochastic sampling of a random variable. (upper left) Plot of the desired density function of equation (15.13); (upper right) Time sequence of sample values generated by the Metropolis algorithm. (lower left) Plot of the proposal density function formed by a mixture of two Gaussians. (lower right) Time sequence of sample values generated by the Hastings-Metropolis algorithm. Notice that adjacent samples generated by the Metropolis algorithm are often the same because the updates are rejected. Alternatively, the proposals of the Hastings-Metropolis are more often accepted, leading to faster convergence of the Markov chain.

where $\sigma = 1/2$. So therefore, we see that the proposal density has the symmetry property required for the Metropolis algorithm.

$$q(w|x^{(k)}) = q(x^{(k)}|w)$$

The acceptance probability is then given by

$$\alpha = \min \left\{ 1, \exp\left\{ -[u(w) - u(x^{(k)})]\right\} \right\} = \min \left\{ 1, \frac{p(w)}{p(x^{(k)})} \right\}.$$  

The time sequence of values generated by the Metropolis algorithm is shown in the upper right hand plot of Figure 15.4. Notice that sequential values are often exactly the same. This is because many of the samples are rejected. Intuitively, this high rejection rate slows down convergence of the Markov chain.

Next, we use the Hasting-Metropolis algorithm to generate samples. For this algorithm, we are not limited to symmetric proposal densities, so we may very roughly approximate the desired distribution of equation (15.13). We do this rough approximation with a mixture of two Gaussian distributions,
one distributed as $N(1, (1/4)^2)$ and a second distributed as $N(-1, (1/4)^2)$. This results in a proposal distribution with the form

$$q(w) = \frac{\pi_1}{\sqrt{2\pi\sigma_w^2}} \exp\left\{-\frac{1}{2\sigma_w^2}(w - 1)^2\right\} + \frac{\pi_2}{\sqrt{2\pi\sigma_w^2}} \exp\left\{-\frac{1}{2\sigma_w^2}(w + 1)^2\right\}$$

where $\sigma_w = 1/4$, $\pi_1 = \frac{\exp(-1)}{1+\exp(-1)}$, and $\pi_2 = \frac{1}{1+\exp(-1)}$. Notice that this density, shown in the lower left corner of Figure 15.4, reasonably approximates the true density. So we would expect that fewer rejections will be needed. For the Hastings-Metropolis algorithm, the acceptance probability is then

$$\alpha = \min\left\{1, \frac{q(x^{(k)})p(w)}{q(w)p(x^{(k)})}\right\}.$$ 

So if $q(x) \approx p(x)$, then the acceptance probability should be $\approx 1$.

The plotted time sequence of the Hastings-Metropolis samples shown in Figure 15.4 supports our intuition. Notice that samples are much less correlated in time than is the case for the Metropolis algorithm; and in particular, there are many fewer examples of repeated samples (i.e., rejected proposals). This means that the Markov chain converges more quickly to its desired distribution, and that samples become approximately independent with fewer time steps of the Markov chain. In practice, this means that fewer samples need to be generated by the Markov chain in order to compute accurate estimates of time averages.

### 15.4 Stochastic Sampling of MRFs

In the previous sections, we have been a bit vague about exactly what $X$ represents. In fact, our theory does not assume a specific form for $X$, but our examples emphasize the case when $X$ is a random variable. While this scalar case is useful, stochastic simulation is more widely used in applications where $X$ is a high dimensional object, such as an MRF. So in this case, the stimulation method produces samples, $X_s^{(k)}$, which vary in both 1-D time, $k$, and space, $s$.

While the methods of Sections 15.2 and 15.3 fully apply when $X$ is an MRF, the particulars of the implementation bear some additional scrutiny.
More specifically, one needs to consider what practical choices exist for the proposal distribution, $q(w|x)$.

When $X$ is high dimensional, it is common to restrict the proposals to changes in only one coordinate of the high dimensional vector. So for example, the proposal distribution is restricted to have the form

$$q_s(w|x) = g_s(w_s|x) \delta(w_r - x_r; r \neq s),$$

where $g_s(w_s|x)$ is the conditional proposal distribution of a single pixel, $W_s$, given the random field $X$. The term $\delta(w_r - x_r; r \neq s)$ in the conditional probability indicates that all other pixels remain the same. So $W_r = X_r$ for $r \neq s$. We refer to this strategy of only replacing a single pixel at a time as a coordinate-wise sampler.

Pseudocode for the corresponding coordinate-wise Hastings-Metropolis algorithm is listed in Figure 15.5. Notice that the pixels in the MRF are first ordered from 0 to $N - 1$, and this ordering is denoted by $\{s_i\}_{i=0}^{N-1}$. Then in a single full iteration through the image, the pixels are visited and updated in this repeating sequence. For this case, the acceptance probability is then given by

$$\alpha = \min \left\{ 1, \frac{q_{s_i}(X^{(k)}|W)}{q_{s_i}(W|X^{(k)})} \exp \left\{ -[u(W) - u(X^{(k)})] \right\} \right\}$$

$$= \min \left\{ 1, \frac{g_{s_i}(X_{s_i}^{(k)}|W)}{g_{s_i}(W_{s_i}|X^{(k)})} \exp \left\{ -[u(W) - u(X^{(k)})] \right\} \right\}. \quad (15.15)$$

Now if the 1-D density function $g_{s}(w_s|x) > 0$ for all $w_s$ and $x$, then on each update it is possible (but perhaps not very likely) to produce any pixel update. And since in one pass through the pixels of $X$, every pixel is updated, then in a single update any new image is possible. This means that the resulting Markov chain is both irreducible and aperiodic Moreover, assuming that the Hastings-Metropolis method is used to accept or reject proposals, then the Markov chain will also be reversible. So we know that with the update of any pixel, $s$, the detailed-balance equations must be satisfied; and therefore, the full-balance equations must also be satisfied by the Gibbs distribution.

$$\frac{1}{\tilde{z}} e^{-u(x^{(k+1)})} = \sum_{x^{(k)} \in \Omega^N} p_s(x^{(k+1)}|x^{(k)}) \frac{1}{\tilde{z}} e^{-u(x^{(k)})} \quad (15.16)$$
### Coordinate-Wise Hastings-Metropolis Algorithm:

Initialize $X^{(0)}$
Select ordering of pixels, $\{s_i\}_{i=0}^{N-1}$
Initialize $k \leftarrow 0$

For $l = 0$ to $L - 1$ { /* Repeat for $L$ full iterations through the MRF */

For $i = 0$ to $N - 1$ {

/*Generate proposal*/
$W \leftarrow X^{(k)}$ /*Initialize proposal*/
$W_{s_i} \sim g_{s_i}(w|X^{(k)})$ /*Proposal for $s_i$th pixel*/

/*Accept or reject proposal*/
$\alpha \leftarrow \min \left\{ 1, \frac{g_{s_i}(X^{(k)}_{s_i}|W)}{g_{s_i}(W_{s_i}|X^{(k)})} \exp\{-[u(W) - u(X^{(k)})]\} \right\}$

$X^{(k+1)} \leftarrow \begin{cases} W \text{ with probability } \alpha \\ X^{(k)} \text{ with probability } 1 - \alpha \end{cases}$

$k \leftarrow k + 1$
}

Figure 15.5: A pseudocode for a Hastings-Metropolis sampler using coordinate-wise updates. The pseudocode performs $L$ complete iterations through the full MRF, and each full iteration performs a single update of each pixel. The resulting Markov chain is nonhomogeneous, but it still converges to the desired Gibbs distribution.

---

However, the analysis of the previous section assumed that the proposal distribution, $q(x^{(k+1)}|x^{(k)})$, is fixed. But in this case, the proposal distribution of equation (15.14) is dependent on $s$, the pixel being updated. Therefore, the resulting Markov chain is not homogeneous. Since it is not homogeneous, it is also unclear as to whether it converges to the desired Gibbs distribution.

Nonetheless, it seems intuitively clear that if the pixels are updated in a fixed and repeated order, then the Markov chain should converge. The following argument confirms this intuition. From Property [13.2] of Chapter [13], we know that the subsampling of any Markov chain produces another Markov chain. So let $X^{(k)}$ be the state after $k$ updates. The subsampled version is then $Y^{(k)} = X^{(Nk)}$ where $N$ is the number of pixels in the image. Then $Y^{(0)} = X^{(0)}$ is the initial state, $Y^{(1)} = X^{(N)}$ is the state after a single update of each pixel, and $Y^{(k)} = X^{(Nk)}$ is the result after $k$ full updates of the state. Now $Y^{(k)}$ is a Markov chain because it is formed by a subsampling of $X^{(k)}$. 
In addition, we know that $Y^k$ forms a homogeneous Markov chain because each full update of the pixels follows the same sequence of transition rules. More formally, we can compute the transition probability for $Y^k$ as

$$\tilde{p}(x^{(N)}|x^{(0)}) = \sum_{x^1 \in \Omega^N} \sum_{x^2 \in \Omega^N} \cdots \sum_{x^{N-1} \in \Omega^N} \prod_{i=0}^{N-1} p_s(x^{(i+1)}|x^{(i)}) ,$$

where $\tilde{p}(x^{(N)}|x^{(0)})$ no longer depends on an index of time. Moreover, since the Gibbs distribution is the stationary distribution of the transitions $p_s(x^{(k+1)}|x^{(k)})$, it must also be the stationary distribution (i.e., solve the full-balance equations) for the transition probabilities $\tilde{p}(x^{(N)}|x^{(0)})$. So we have that

$$\frac{1}{z} e^{-u(y^{(k+1)})} = \sum_{y^k \in \Omega^N} \tilde{p}(y^{(k+1)}|y^{(k)}) \frac{1}{z} e^{-u(y^{(k)})} .$$

Finally, if $g_s(w_s|x) > 0$ then in one full iteration of the algorithm we can get from any state $Y^{(k)}$ to any other state $Y^{(k+1)}$. So therefore, the Markov chain $Y^{(k)}$ must be irreducible and aperiodic.

So now we have all the ingredients required for convergence of the ergodic Markov chain $Y^{(k)}$ to the Gibbs distribution: 1) The Markov chain is stationary; 2) it has a finite state; 3) it is homogeneous; 4) it is irreducible; 5) it is aperiodic; and 6) the Gibbs distribution solves the full-balance equations of the Markov chain.

With this, we can now state a theorem about the convergence of the coordinate-wise Hastings-Metropolis sampler.

**Theorem 15.4.1. Convergence of Coordinate-Wise Hastings-Metropolis Algorithm**

Let $X^{(k)} \in \Omega^N$ be the $k^{th}$ update of a coordinate-wise Hastings-Metropolis algorithm sampling from the Gibbs distribution $p(x) = \frac{1}{z} e^{-u(x)}$ with $\Omega$ a finite discrete set. (See Figure [15.5]) Furthermore, assume that the coordinate-wise proposal distribution is strictly positive, i.e., for all $w_s \in \Omega$, $g_s(w_s|x) > 0$. Then $X^{(k)}$ is a Markov chain, which reaches the stationary distribution

$$\lim_{k \to \infty} P\{X^{(k)} = x\} = \frac{1}{z} e^{-u(x)} .$$

This theorem, which is based on the convergence Theorem [13.4.1] of Section [13.4], ensures that the coordinate-wise Hastings-Metropolis algorithm
always converges to the desired result. However, in some cases, this convergence can be very slow.

Moreover, this theorem also guarantees convergence of the basic-Metropolis algorithm, since that is a special case. Another special case, that is widely used in practice, is the so-called Gibbs sampler presented in the following section.

15.5 The Gibbs Sampler

<table>
<thead>
<tr>
<th>Gibbs Sampler Algorithm:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize $X^{(0)}$</td>
</tr>
<tr>
<td>Select ordering of pixels, ${s_i}_{i=0}^{N-1}$ /* Can use raster order */</td>
</tr>
<tr>
<td>Initialize $k \leftarrow 0$</td>
</tr>
<tr>
<td>For $l = 0$ to $L - 1$ {</td>
</tr>
<tr>
<td>For $i = 0$ to $N - 1$ {</td>
</tr>
<tr>
<td>/<em>Generate update for $s_i^{th}$ pixel</em>/</td>
</tr>
<tr>
<td>$X^{(k+1)} \leftarrow X^{(k)}$ /<em>Initialize update</em>/</td>
</tr>
<tr>
<td>$W_{s_i} \sim g_{s_i}(w_{s_i} \mid X^{(k)}_{\partial s_i})$ /* Generate sample from conditional distrib.*/</td>
</tr>
<tr>
<td>$X^{(k+1)}<em>{s_i} \leftarrow W</em>{s_i}$ /* Replace pixel with generated sample*/</td>
</tr>
<tr>
<td>$k \leftarrow k + 1$</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

Figure 15.6: The Gibbs sampler algorithm is a special case of the coordinate-wise Hastings-Metropolis algorithm when the proposal distribution for each pixel is just its conditional distribution given its neighbors, i.e., $X^{(k+1)}_{s_i} \sim p_{s_i}(x_{s_i} \mid X^{(k)}_{\partial s_i})$. In this case, the acceptance probability is always $\alpha = 1$; so all proposals are accepted.

The **Gibbs sampler** is an important special case of the coordinate-wise Hastings-Metropolis sampler that was first proposed by Geman and Geman [28] and has since become widely used due to its ease of implementation and relatively fast convergence for simple distributions. With the Gibbs sampler, the proposal distribution is taken to be the conditional distribution of the pixel $X_s$ given its neighbors under the Gibbs distribution.
Assume that $X$ is an MRF with a Gibbs distribution of the form
\begin{equation}
p(x) = \frac{1}{z} \exp \left\{ -u(x) \right\}
= \frac{1}{z} \exp \left\{ -\sum_{c \in C} V_c(x_c) \right\},
\end{equation}
where $u(x)$ is the energy function and $V_c(x_c)$ are the potential functions for clique $c$. If we again define $C_s$ from equation (14.2) to be the subset of all cliques containing $x_s$, then we know from equation (14.3) that the conditional distribution of a pixel, $x_s$, given all other pixels has the form
\begin{equation}
P\{X_s = x_s | X_r \neq s\} = \frac{\exp \{ -u_s(x_s | x_{\partial s}) \}}{\sum_{x_s \in \Omega} \exp \{ -u_s(x_s | x_{\partial s}) \}}
\end{equation}
where
\begin{equation}
u_s(x_s | x_{\partial s}) = \sum_{c \in C_s} V_c(x_c).
\end{equation}

So then for a Gibbs sampler, the key idea is to use the conditional distribution of a pixel as the proposal distribution. More specifically, the proposal distribution is given by
\begin{equation}
g_s(x_s | x_{\partial s}) = P\{X_s = x_s | X_i = x_i \ i \neq s\} = \frac{\exp \{ -u_s(x_s | x_{\partial s}) \}}{\sum_{x_s' \in \Omega} \exp \{ -u_s(x_s' | x_{\partial s}) \}}.
\end{equation}

In this case, the acceptance probability can be computed in closed form to be
\begin{equation}
\alpha = \min \left\{ 1, \frac{g_s(X_s^{(k)} | W_{\partial s})}{g_s(W_s | X_{\partial s}^{(k)})} \exp \left\{ -[u(W) - u(X_s^{(k)})] \right\} \right\}
= \min \left\{ 1, \frac{g_s(X_s^{(k)} | W_{\partial s})}{g_s(W_s | X_{\partial s}^{(k)})} \exp \left\{ -u(W_s | X_{\partial s}^{(k)}) \right\} \right\}
= \min \{1, 1\} = 1.
\end{equation}

\footnote{Without loss of generality, $X$ can be assumed to be an MRF because we can always take $\partial s = \{ r : r \neq s \}$.}
So the acceptance probability for the Gibbs sampler is always 1. Without rejections, the algorithm becomes the simple sequential replacement of pixels from their assumed conditional distribution.

For example, with the Ising model of Section 14.3, the conditional distribution of a pixel given its neighbors becomes

\[ g_s(x_s|x_{\partial s}) = \frac{\exp\left\{ -\beta \sum_{r \in \partial s} \delta(x_s \neq x_r) \right\}}{\sum_{x_s=0}^{M-1} \exp\left\{ -\beta \sum_{r \in \partial s} \delta(x_s \neq x_r) \right\}}. \]  \hspace{1cm} (15.18)

So for this case, the Gibbs sampler would simply replace each pixel in sequence with a new pixel drawn from this conditional distribution. An important advantage of the Gibbs sampler is that, since all proposals are accepted, it tends to achieve faster convergence. Intuitively, each pixel is replaced with a sample having the desired distribution. So it is quite reasonable that repeated updating of pixels with this “correct” distribution should converge to the entire random field having the correct distribution.

Figure 15.7 shows the results of an example in which a Gibbs sampler is used to generate pseudo-random samples from the distribution of a $16 \times 16$ Ising model with circular boundary conditions and an inverse temperature parameter of $\beta = 1.0$. So this example corresponds to the conditional distribution of equation (15.18), but with a 4-point neighborhood and $M = 2$. The figure shows 4 independent runs of the simulation, each with an independent initial condition of i.i.d. binary random variables with equal probability of 1 (white) and $-1$ (black). The columns are the results after 0, 10, 50, and 100 full iterations through the image. Notice that after a large number of iterations, the result becomes either mostly all 1’s or mostly all $-1$’s. This happens because $\beta$ is above the critical value; so we know that the distribution of the Ising model is bi-modally disturbed, with one mode consisting of mostly all 1’s, and the other mode consisting of mostly all $-1$’s.

For this example, the Gibbs sampler does a good job of generating representative samples from the distribution of the Ising model, but as the size of the image grows larger, or as the value of $\beta$ grows larger, one can easily imagine that the Gibbs sampler can become “trapped” in a deep local maximum of the distribution. In fact, while the Gibbs sampler is guaranteed to converge, in many practical cases its convergence can be extremely slow. In some cases, the convergence is so slow that no practical computer could generate the desired result. We will look at this issue in more depth in the
Figure 15.7: A Gibbs sampler simulation of an Ising model with inverse temperature parameter of $\beta = 1.0$ for a $16 \times 16$ binary array using circular boundary conditions. Each row represents an independently run experiment using a different seed for the pseudo-random number generator, and the columns are the results after 0, 10, 50, and 100 full iterations through the image. Notice that after a large number of iterations, the result becomes either mostly all 1’s or mostly all −1’s because $\beta$ is above the critical value.

following chapter.
1. Let $W = X^k + Z$ where $Z$ is independent of $X^k$ with density $p_z(z)$. Show that the proposal density $q(w|x)$ obeys the symmetry condition of equation (15.4).

2. The following problem verifies some properties of the Metropolis simulation algorithm.
   a) Show that the transition probabilities for the Metropolis algorithm given by equation (15.6) are correct.
   b) Justify each line in the proof of equation (15.7).
   c) Use the symmetry of equation (15.8) to show that it also holds in case 2 when $u(x') > u(x)$ and $x' \neq x$.
   d) Show that equation (15.8) also holds in case 3 when $x' = x$.

3. Justify each line in the proof of equation (15.11). Then prove the result of equation (15.12) using the three cases given in the text.

4. Show that when the proposal distribution is taken to be
   \[
   q(w|x^k) = \frac{1}{z} \exp\{-u(w)\},
   \]
   then the acceptance probability is 1, and the algorithm converges to the Gibbs distribution in one step.

5. Consider the random variable $X$ with density
   \[
   p(x) = \frac{1}{\sigma z(p)} \exp\left\{-\frac{1}{p\sigma^p}|x|^p\right\},
   \]
   with $p = 1.2$ and $\sigma = 1$. Consider the case of a Metropolis simulation algorithm for sampling from the distribution of $p(x)$ with the proposals generated as $W \leftarrow X^k + Z$ where $Z \sim N(0, 1)$.
   a) Plot the density function for $p(x)$.
   b) Derive an expression for the proposal distribution $q(w|x)$, and show that the proposal distribution obeys the symmetry condition given by $q(w|x) = q(x|w)$.
   c) Derive an expression for the acceptance probability $\alpha$. 
d) Write a Matlab program to generate 100 Metropolis samples from an initial condition of \( X^0 = 0 \), and plot your output.

6. Consider the positive random variable \( X \) with density

\[
p(x) = \frac{1}{\sigma z(p)} \exp \left\{ -\frac{1}{p\sigma^p |x|^p} \right\},
\]

with \( p = 1.2 \) and \( \sigma = 1 \). Consider the case of a Hastings-Metropolis simulation algorithm for sampling from the distribution of \( p(x) \) with the proposals generated as \( W \sim q(w) \) where

\[
q(w) = \frac{1}{2\gamma} \exp \left\{ -\frac{1}{\gamma} |w| \right\},
\]

where \( \gamma = 1 \).

a) Plot the density function for \( p(x) \) and the proposal density \( q(w) \). How do they compare?

b) Derive an expression for the acceptance probability \( \alpha \).

c) Write a Matlab program to generate 100 Hastings-Metropolis samples from an initial condition of \( X^0 = 0 \), and plot your results.

d) Compare your results to the results of Problem 5 above.

7. Let \( X_n \) be a sequence of multivariate random vectors that form a homogeneous Markov Chain. More specifically, for each \( n \), let \( X_n = (X_{n,0}, X_{n,1}, \cdots, X_{n,M-1}) \) where \( X_{n,m} \in \{0, \cdots, K-1\} \).

Furthermore, let \( p(x) > 0 \) be any probability density function defined over the set \( x \in \{0, \cdots, K-1\}^M \), and let \( p_m(x_m|x_i \text{ for } i \neq m) > 0 \) be its associated conditional density functions.

Then each new state is generated using the following three step procedure:

**Step 1:** Generate a uniformly distributed random variable \( J \) on the set \( \{0, \cdots, M-1\} \).

**Step 2:** Generate an independent random variable \( W \), distributed according to \( W \sim p_J(x_J|X_i \text{ for } i \neq J) \).
Step 3: For $i \neq J$, set $X_{n,i} \leftarrow X_{n-1,i}$; and for $i = J$, set $X_{n,i} \leftarrow W$.

a) Show that the Markov chain has a finite number of states.

b) Show that the Markov chain is irreducible.

c) Show that the Markov chain is aperiodic.

d) Prove that Markov chain is ergodic with asymptotic distribution $p(x)$.

e) Intuitively, why does this make sense?
Chapter 16

Bayesian Segmentation

Up until this point, our primary focus has been on the use of MRF priors in applications where the unknown image, $X$, is continuously valued. A partial exception in Chapter 13 was the estimation of discrete Markov chain states using a doubly-stochastic hidden Markov model.

In this chapter, we study the application of model-based methods to the estimation of discrete random-fields, $X$, from observation data. In many applications, this is essentially equivalent to segmentation of an image, with the discrete state of each pixel in $X$ representing a class of the segmentation. To do this, we will use the tools introduced in Chapter 14 to model general discrete MRFs along with the computational techniques of Chapter 15 to compute estimates of the discrete image $X$.

An important point to remember is that the methods of this chapter not only apply to the traditional segmentation of images. They also can be applied to the reconstruction of images when the images have discrete properties. This is extremely important when imaging 3-D objects composed of a small number of discrete materials since the segmentation into these discrete regions can be a powerful constraint.

16.1 Framework for Bayesian Segmentation

Figure 16.1 graphically illustrates the basic framework for MAP segmentation of an image. The unknown segmentation is represented by $X$, a random field taking on discrete values $X_s \in \Omega = \{0, \cdots, M - 1\}$ for $s \in S$. Each class is denoted by a discrete value 0 through $M - 1$, and each pixel is assumed
to take on one of these \( M \) discrete classes. Typically, the random field of classes, \( X \), is assumed to be a discrete MRF with a Gibbs distribution of form in equation (14.1). Then by estimating \( X \), we then recover the unknown segmentation of the observed image \( Y \).

Our estimate of \( X \) is based on the observed data \( Y \), through a forward model, \( p(y|x) = P\{Y \in dy|X = x\} \), and a prior model, \( p(x) = P\{X = x\} \). In a typical case, \( p(x) \) is a Gibbs distribution for a discrete MRF, and \( p(y|x) \) is formed by a local dependency of each observed pixel \( Y_s \) on its unknown class, \( X_s \). So for example, if each observed pixel, \( Y_s \), is only dependent on its associated class label, \( X_s \), then we have

\[
p_{y|x}(y|x) = \prod_{s \in S} p_{y_s|x_s}(y_s|x_s)
\]

(16.1)

where \( p_{y_s|x_s}(y_s|x_s) \) is the conditional distribution of \( Y_s \) given \( X_s \). In this case, we say that the likelihood has **product form**, and the negative log likelihood used in MAP estimation can be written as a sum of independent terms,

\[
- \log p_{y|x}(y|x) = \sum_{s \in S} l(y_s|x_s)
\]

(16.2)

where \( l(y_s|x_s) = - \log p_{y_s|x_s}(y_s|x_s) \).

While assuming this type of product form can be extremely useful, it is also a strong assumption because it presumes that neighboring pixels are
conditionally independent given the class labels $X$. In natural images, there are usually spatial correlations between nearby pixels of the same class; so this type of model does not account for such spatial dependencies. A partial solution to this problem is to let each observed pixel, $Y_s$, be an local feature vector extracted from the image. Of course, this is also approximate since neighboring feature vectors will likely share information and be conditionally dependent, but nonetheless, the approach can be effective.

The form of equation (16.1) allows for a wide range of choices in the distribution of $p_{y_s|x_s}(y_s|x_s)$. Just about any distribution parameterized by $\phi$ can be used. In this case, each of the $M$ classes can be specified by a different choice of the parameter $\{\phi_m\}_{m=0}^{M-1}$, and the negative log likelihood is given by

$$l(y_s|x_s) = -\log p_{y_s|x_s}(y_s|\phi_{x_s}) .$$

For example, if each class is assumed to be conditionally Gaussian, then

$$l(y_s|x_s) = \frac{1}{2\sigma^2_{x_s}} (y_s - \mu_{x_s})^2 + \frac{1}{2} \log (2\pi\sigma^2_{x_s}) ,$$

and each pixel $Y_s$ is a conditionally independent Gaussian random variable with mean $\mu_{x_s}$ and variance $\sigma^2_{x_s}$.

Another common choice is to assume that each pixel $Y_s$ is conditionally distributed with a Gaussian mixture from Section 12.2. This has the advantage that each of the $M$ Gaussian mixture distributions can be fitted to the empirical distribution of each class. Of course, the parameters of a Gaussian mixture can be fitted using ML estimation with the EM algorithm of Section 12.4. In this case, each parameter vector $\phi_m$ parameterizes the Gaussian mixture distribution of the $M^{th}$ class.

More specifically, we can model $Y_s$ as a p-dimensional Gaussian mixture given the class label $X_s$. The parameters of this model can be estimated using the EM algorithm as described in Section 12.7.3. So we have that for a $J$ component Gaussian mixture, the parameter to be estimated is

$$\phi_m = \{\pi_{m,j}, \mu_{m,j}, B_{m,j}\}_{j=0}^{J-1} ,$$

where $\{\pi_{m,j}\}_{j=0}^{J-1}$ are the $J$ component weightings that sum to 1, $\{\mu_{m,j}\}_{j=0}^{J-1}$ are the component means, and $\{B_{m,j}\}_{j=0}^{J-1}$ are the component precision (i.e., inverse covariance) matrices. Then the conditional distribution of $Y_s$ given
\( X_s \) is

\[
p_{y_s|x_s}(y_s|m, \phi) = \sum_{j=0}^{J-1} \frac{\pi_{m,j} |B_{m,j}|^{1/2}}{\sqrt{2\pi}} \exp \left\{ -\frac{1}{2} (y_s - \mu_{m,j})^t B_{m,j} (y_s - \mu_{m,j}) \right\},
\]

and the negative log likelihood is given by

\[
l(y_s|m, \phi) =
- \log \left( \sum_{j=0}^{J-1} \frac{\pi_{m,j} |B_{m,j}|^{1/2}}{\sqrt{2\pi}} \exp \left\{ -\frac{1}{2} (y_s - \mu_{m,j})^t B_{m,j} (y_s - \mu_{m,j}) \right\} \right).
\]

Once forward and prior models have been chosen, we can apply some type of Bayesian estimator to determine \( X \) from \( Y \). The general approach, described in Section 2.3.2, is to select a cost function, \( C(x, \hat{x}) \), and then choose the estimator \( \hat{X} = T(Y) \) that minimizes the expected cost.

\[
T^* = \arg \min_T \mathbb{E}[C(X, T(Y))]
\]

If the cost is chosen to be \( C(x, \hat{x}) = 1 - \delta(x - \hat{x}) \), then the optimal estimator is the MAP estimate given by

\[
\hat{X} = T^*(Y) = \arg \max_{x \in \Omega^N} \{ p_{x|y}(x|Y) \}
= \arg \min_{x \in \Omega^N} \{ -\log p_{y|x}(Y|x) - \log p(x) \}.
\]

However, the cost function of the MAP estimate is difficult to justify since it is only 0 when there is an exact match between the true segmentation \( X \) and the estimate \( \hat{X} \). This is a very conservative criteria since it means that having one erroneous pixel classification is no worse than having every pixel wrong. Nonetheless, the MAP estimate is widely used because it can be implemented with numerical optimization, and it often yields good results.

A more compelling cost criteria actually counts the number of classification errors in the image, rather than simply checking to see if the classification is perfect. This cost function is given by

\[
C(x, X) = \sum_{s \in S} \delta(x_s \neq X_s),
\]
and minimization of this cost function results in the so-called maximizer of the posterior marginals (MPM) estimator \[45\] given by

\[
\hat{X}_s = \arg \max_{x_s} p_{x_s|Y}(x_s|Y). \tag{16.6}
\]

This MPM estimator has the property that it minimizes the number of misclassified pixels in the image, which is quite reasonable in most applications, but it has the disadvantage that it requires the computation of an \((N-1)\)-dimensional integral or expectation.

\[
\hat{X}_s = \arg \max_{x_s} \mathbb{E}[p_{y|x}(Y|x_s, X_r \neq s)p(x_s, X_r \neq s)|Y] \tag{16.7}
\]

While computing this high-dimensional integral for each pixel might seem impossible, it can be done using the stochastic sampling techniques of Chapter \[15\] as we will describe in Section \[16.1.2\].

In the following sections, we will introduce some of the methodologies for computing Bayesian estimates of discrete random fields, \(X\), with a structure as shown in Figure \[16.1\]. In particular, we will present basic methods for computing both the MAP and MPM estimates of equations (16.5) and (16.7).

### 16.1.1 Discrete Optimization for MAP Segmentation

In order to illustrate basic methods for optimization of the MAP cost function, we start by assuming that the image has a log likelihood with a product-form as in equation (16.1) and a prior model based on an M-level discrete MRF. Then the negative log likelihood is given by

\[
- \log p_{y|x}(y|x) = - \sum_{s \in S} \log p_{y_s|x_s}(y_s|x_s) \\
= \sum_{s \in S} l(y_s|x_s),
\]

where \(l(y_s|x_s) = - \log p_{y_s|x_s}(y_s|x_s)\) is the negative log likelihood for each pixel, and the negative log prior term is given by

\[
- \log p(x) = \beta \sum_{\{s,r\} \in C} \delta(x_r \neq x_s) + \text{const}
\]
where \( \mathcal{C} \) contains all pairwise cliques in an 8-point neighborhood. Using these two models, the MAP estimate is given by

\[
\hat{x} = \arg \min_{x \in \Omega^N} \left\{ -\log p_y(x|y) - \log p(x) \right\} = \arg \min_{x \in \Omega^N} \left\{ \sum_{s \in S} l(y_s|x_s) + \beta \sum_{\{s,r\} \in \mathcal{C}} \delta(x_r \neq x_s) \right\}.
\]  

(16.8)

In general, optimization problems with the form of equation (16.8) are very difficult to solve. Since they are discrete, they certainly cannot be convex. In fact, even the definition of convexity no longer applies since it requires the function to be defined along any line connecting two inputs values to the function, and a linear combination of two discrete variables is not necessarily discrete.

In general this optimization problem will have exponential complexity. However, in the special case when \( M = 2 \), this type of optimization can be solved exactly in polynomial time by using graph cut algorithms based on the max-flow min-cut theorem [34]. This class of graph cut algorithms can be solved in polynomial time by the Ford-Fulkerson algorithm [22]. In fact, this approach works for any problem with the form of equation (16.8) when \( M = 2 \). Moreover, it even holds when the neighborhoods have arbitrary structure and the weights of the potential function are space varying. So for example, it works for binary segmentation using the \( M \)-level discrete MRF of equation (14.17) with \( M = 2 \).

However, the exact graph-cut solution does not work when \( M > 2 \), and it generally only works when the log likelihood of the data has a simple product form of equation (16.1). In particular, the product form is often not a realistic assumption when pixels are conditionally dependent or depend on the class label of nearby pixels. Nonetheless, graph-cut algorithms have become very popular in image segmentation, so these can be a very useful technique [16].

So absent an exact algorithm for minimization of equation (16.8), we must resort to some approximate method of minimization. Early on, there were many attempts to use dynamic programming to solve this MAP segmentation problem in much the same way that dynamic programming can be used to solve the MAP estimation problem for HMMs. However, dynamic programming does not work in 2-D because the state variable must have the dimension
of an entire row or column of the image, which makes the complexity exponential. Nonetheless, various approximation methods have been proposed for doing approximate MAP segmentation by, for example, processing a small number of image rows at a time \cite{18, 71}.

Perhaps the most common approach to minimization of this cost function is coordinate-wise optimization similar to ICD. For discrete random fields, $X$, this approach is often called **iterated conditional modes (ICM)** $\cite{8}$. ICM works by iteratively replacing each pixel’s class with the class that maximizes the mode of the conditional distribution $p(x_s|y, x, r \neq x)$. This can be computed as

$$
\hat{x}_s \leftarrow \arg \max_{x_s} p(x_s|y, x_r r \neq s)
= \arg \min_{x_s} \left\{ -\log p_{y|x}(y|x_s, x_r r \neq s) - \log p(x_s, x_r r \neq s) \right\}
= \arg \min_{x_s} \left\{ l(y_s|x_s) - \log p(x_s, x_r r \neq s) \right\}
= \arg \min_{x_s} \left\{ l(y_s|x_s) + \beta \sum_{\{i,j\}\in C} \delta(x_i \neq x_j) \right\}
= \arg \min_{x_s} \left\{ l(y_s|x_s) + \beta \sum_{r \in \partial s} \delta(x_s \neq x_r) \right\}.
$$

So with this, we see that replacing each pixel with its conditional mode is exactly equivalent to performing an ICD update, i.e., sequentially replacing each pixel with the value that maximizes the MAP cost function. So ICD and ICM are equivalent for discrete random fields, $X$.

Figure 16.2 provides a pseudocode specification of the ICM algorithm. The segmentation is initialized with the ML estimate of the class of each pixel. This is simply the class that maximizes the likelihood for each observation $Y_s$. Then for each iteration of the algorithm, every pixel is visited and replaced with the class that maximizes the MAP cost function. This can be done with little computation since each pixel is only dependent on the local negative log likelihood, $l(y_s|x_s)$, and the class of its neighbors. Each pixel replacement monotonically decreases the MAP cost function, and the MAP cost function is bounded below, so this sequence is guaranteed to converge to a local minimum of the cost function in a finite number of steps. When this happens, each
new pixel replacement remains the same and the iterations stop. However, while ICM can be a simple and effective algorithm for MAP segmentation, it does have the problem that it is easily trapped in local minimum of the MAP cost function, particularly when the signal-to-noise ratio is low.

![ICM Algorithm:](image)

Figure 16.2: A pseudocode specification of the ICD algorithm for computing an approximate MAP estimate of a discrete random field $X$. This algorithm is typically used to segment images from observed data $Y$.

One approach to avoiding local minima in optimization is simulated annealing (SA). The concept of SA is to use the stochastic sampling methods of Chapter 15 to generate samples from a distribution whose maximum also minimizes the MAP cost function. So for our problem, the MAP cost function is given by

$$u(x) = \sum_{s \in S} l(y_s|x_s) + \beta \sum_{\{s,r\} \in C} \delta(x_r \neq x_s), \quad (16.9)$$

where we use the notation $u(x)$ in order to draw an analogy with the energy of a Gibbs distribution. In fact, it is easily shown that the posterior distribution of the segmentation, $p_{x|y}(x|y)$, is given by the following Gibbs distribution

$$p_T(x|y) = \frac{1}{z} \exp \left\{ -\frac{1}{T} u(x) \right\},$$

when the temperature $T = 1$. However, as the temperature is reduce so that

---

1 An interesting dilemma occurs when the $M$ possible values of the cost functions are equal for a pixel replacement. If the tie-breaking algorithm is deterministic, then convergence of ICM is guaranteed. However, if the tie-breaking algorithm is, for example, stochastic, then it is possible to have limit-cycles that do not allow convergence of the state, $x$. In this case, the pixel choices can cycle between states that have equal values of the MAP cost function. One such tie-breaking procedure that guarantees convergence is to only change the current pixel value when the new class strictly decreases the MAP cost function.
\( T \leq 1 \), samples from this distribution become more concentrated about the MAP estimate of \( X \).

To better illustrate this point, consider the MAP estimate

\[
\hat{x} = \arg\min_{x \in \Omega^N} u(x) ,
\]

and let \( x \neq \hat{x} \) be any other value of \( x \). Then we can see that as the temperature goes to zero, the likelihood ratio between the probability of the MAP estimate and any other state goes to infinitely.

\[
\lim_{T \downarrow 0} \frac{p_T(\hat{x})}{p_T(x)} = \lim_{T \downarrow 0} \exp \left\{ \frac{1}{T} (u(x) - u(\hat{x})) \right\} = \lim_{T \downarrow 0} \exp \left\{ \frac{1}{T} (u(x) - \min_{x \in \Omega} u(x)) \right\} = \infty
\]

Moreover, since \( p_T(\hat{x}_{MAP}) \leq 1 \), we then know that for \( x \neq \hat{x} \) then

\[
\lim_{T \downarrow 0} p_T(x) = 0 .
\]

So if the MAP estimate, \( \hat{x} \), is unique, then

\[
\lim_{T \downarrow 0} p_T(\hat{x}) = 1 .
\]

In other words, as the temperature goes to zero, any stochastic sample from the distribution \( p_T(x) \) will be the MAP estimate with probability one.

So the solution might seem simple. Just set the temperature, \( T \), to some very small value, and use the stochastic sampling methods of Chapter 15 to generate a sample from the distribution \( p_T(x) \) using, say, the Metropolis-Hastings sampler or perhaps the Gibbs sampler. However, the problem with this approach is that as the temperature becomes small, the convergence of the Metropolis-Hastings algorithm becomes very, very slow. This is because the probably of transitioning between some states in the MC becomes very small, effectively violating the assumption of irreducibility for the reversible MC. That is to say, that when \( T \) is very small, then it become very improbable to transition between some states, so the MC is, for all practical purposes, no longer irreducible; and therefore, it is no longer ergodic.

\(^2\)Here we are assuming that \( \hat{x} \) represents an equivalence class of states that all minimize the MAP cost function.
The partial solution to this problem is to start the temperature, $T$, at a large value, and then allow its value to decrease slowly as samples from the Metropolis-Hastings algorithm are generated. This means that the resulting MC is no longer homogeneous. Nonetheless, it is still possible to show convergence of this nonhomogeneous MC to the MAP estimate, $\hat{x}$, under the proper conditions.

In practice, the Gibbs sampler is typically used in this application since then there is no need to do rejections. Now in order to implement the Gibbs sampler, we must compute the conditional distribution of each pixel given all the other pixels in the image. For the distribution, $p_T(x)$, it is easily shown that this conditional distribution is given by

$$p_T(x_s|y, x_r \neq s) = \frac{1}{z} \exp \left\{ \frac{1}{T} u(x_s|x_r \neq s) \right\}$$

$$= \frac{\exp \left\{ \frac{1}{T} u(x_s|x_r \neq s) \right\}}{\sum_{m=0}^{M-1} \exp \left\{ \frac{1}{T} u(m|x_r \neq s) \right\}}, \quad (16.10)$$

where

$$u(x_s|x_r \neq s) = l(y_s|x_s) + \beta \sum_{r \in \partial s} \delta(x_r \neq x_s).$$

The SA algorithm is then shown in Figure 16.3. First a annealing schedule denoted by $T_k$ is chosen to be a decreasing sequence of temperatures. In each iteration of the algorithm, each pixel is replaced by a new independent random variable with a distribution take from equation (16.10). As the temperature decreases, samples from the distribution should become clustered more closely to the MAP estimate, $\hat{x}$. However, if the temperature decreases too quickly, then one might expect that the MC will not achieve its desired stationary distribution at each temperature and convergence will not occur.

Figure 16.4 illustrates the value of SA for a simple example. The top row shows the result of different numbers of full iterations of ICD, and the bottom row shows the same thing for the SA algorithm. Notice that the SA result is better because it does not become trapped in a local minimum of the cost function. This can often happen when the two classes have very similar distributions, as is the case in this example.

So the question remains, what annealing schedule, if any, ensures convergence of the SA algorithm to the MAP estimate? In fact, this question was
SA Algorithm:

Initialize with ML estimate $x_s \leftarrow \arg \min_{0 \leq m < M} l(y_s|m)$

Select an annealing schedule: Decreasing sequence $T_k$

For $k = 1$ to $\infty$

For each pixel $s \in S$

Compute the conditional energy

$$u(x_s|x_r r \neq s) = l(y_s|x_s) + \beta \sum_{r \in \partial s} \delta(x_r \neq x_s)$$

Generate a new pixel

$$x_s \sim \frac{\exp \left\{ \frac{1}{T_k} u(x_s|x_r r \neq s) \right\}}{\sum_{m=0}^{M-1} \exp \left\{ \frac{1}{T_k} u(m|x_r r \neq s) \right\}}$$

Figure 16.3: A pseudocode specification of the SA algorithm for computing an approximate MAP estimate of a discrete random field $X$. The SA algorithm has the advantage that it can avoid local minima in the MAP cost function, but at the cost of often substantial additional computation.

answered by Geman and Geman in their classic paper [28]; however, the answer is perhaps still unsatisfying. Let $N$ be the number of pixels in $X$, and define $\Delta = \max_x u(x) - \min_x u(x)$. Then an annealing schedule with the form

$$T_k = \frac{N\Delta}{\log(k + 1)}$$

produces an sequence of Gibbs samples $X^{(k)}$ that converge to the MAP estimate $\hat{x}$ almost surely [28]. However, the problem is that this annealing schedule is absurdly slow. In fact, for this reason has been said that “SA minimization is only slightly worse than exhaustive search.” For example, if $N = 10,000$ and $\Delta = 1$, then in order to achieve a temperature of $T_k = 1$ corresponding to simply sampling form the posterior distribution requires a number of simulation steps of $k \geq e^{10,000} - 1$. This is completely impractical!

So in practice, people use SA but with much faster annealing schedules that allow for some type of reasonable tradeoff between speed and accuracy. For example, a more typical annealing schedule that achieves a reasonable

\footnote{This insightful statement was reported to have been made by Allan Willsky.}
approximate solution is given by
\[ T_k = T_0 \left( \frac{T_K}{T_0} \right)^{k/K}, \]
where \( T_K \) is the selected temperature after \( K \) iterations and \( T_0 \) is the initial temperature.

### 16.1.2 Stochastic Integration for MPM Segmentation

As we discussed in Section 16.1.1, the MPM estimate of a discrete random field can be superior to the MAP estimate because it actually minimizes the probability of error in the estimated segmentation. However, the drawback of the MPM estimate is the potentially high cost of computation. More specifically, the MPM estimate requires that we compute the conditional distribution of a pixel, \( X_s \), given all the data \( Y \); and this, in turn, requires integration over the \( N - 1 \) variables, \( X_r \) for \( r \neq s \). Of course, exact integration is not possible for any practical value of \( N \); but in the following section, we
will see that it is often possible to get a sufficiently accurate approximation to this integral by using the simulated annealing approach of the previous section.

To do this, we first generate a set of \( L \) discrete images, each generated from the conditional posterior distribution, \( p_{x|y}(x|Y) \). So more specifically, let \( X^{(l)} \sim p_{x|y}(x|Y) \) for \( l = 0, \cdots, L-1 \) be a set of \( L \) images each sampled from the posterior distribution \( p_{x|y}(x|Y) \). Then define the normalized histogram for the pixel \( X_s \) as

\[
h_s(x_s) = \frac{1}{L} \sum_{l=0}^{L-1} \delta(X_s^{(l)} - x_s).
\]

Then \( h_s(x_s) \) is essentially an empirical estimate of the conditional distribution of \( X_s \) given \( Y \). In particular, we also know that the conditional mean of each entry in the sum is given by

\[
\mathbb{E}\left[\delta(X_s^{(l)} - x_s)|Y\right] = p_{x_s|y}(x_s|Y).
\]

Then using the law of large numbers, we know that the following sequence of equalities hold,

\[
\lim_{L \to \infty} h_s(x_s) = \lim_{L \to \infty} \frac{1}{L} \sum_{l=0}^{L-1} \delta(X_s^{(l)} - x_s) = \mathbb{E}\left[\delta(X_s^{(l)} - x_s)|Y\right] = p_{x_s|y}(x_s|Y),
\]

where the second equality results from the strong law of large numbers and the fact that the sample images, \( X^{(l)} \), are assumed to be independent samples from the posterior distribution.

So with this, we can use this empirical estimate of the posterior to compute an approximate MPM estimate of \( X \) as

\[
\hat{x}_s = \arg \max_{x_s} p_{x_s|y}(x_s|Y) \approx \arg \max_{x_s} h_s(x_s).
\]

In practice, to do this one need only generate \( L \) samples of the posterior image, and then for each pixel, create a histogram, and select the class that maximizes the histogram.
So in order to generate the samples from the posterior, we can again use the stochastic sampling algorithms of Chapter 15 to generate the images $X^{(l)}$. To do this, we adopt the energy function corresponding to the MAP cost function of equation (16.9).

$$u(x) = -\log p(y|x) - \log p(x) + \text{const}$$

$$= \sum_{s \in S} l(y_s|x_s) + \beta \sum_{\{s,r\} \in C} \delta(x_r \neq x_s)$$

Then we sample from the Gibbs distribution

$$p_T(x|y) = \frac{1}{z} \exp \left\{ -\frac{1}{T} u(x) \right\},$$

with a temperature of $T = 1$. In fact, when $T = 1$, the Gibbs distribution is exactly equal to the posterior distribution $p_{x|y}(x|y)$, and samples generated using the Hastings-Metropolis algorithm will be approximately independent samples from the posterior distribution. Of course, this assumption of independent samples from the posterior distribution in turn depends on the assumption that the MC is run long enough to achieve its steady-state ergodic distribution. So in practice, there is a tradeoff between speed and accuracy in the MPM algorithm.

Figure 16.5 provides a pseudocode specification of the MPM algorithm. As the number of samples, $L$, is increased, the accuracy of the method improves, but at the cost of additional computation. Also, each sample, $X^{(l)}$, is assumed to be an independent sample from the posterior distribution. In order to ensure this, the number of stochastic samples used in the Hastings-Metropolis algorithm must be increased. So, generally speaking, the MPM algorithm can be computationally expensive to implement, but it can also achieve high quality segmentations.

### 16.1.3 Multiscale Approaches to MAP Segmentation

The SA method is useful for experimentation, but in practice, it is usually necessary to have a faster algorithm in real applications. One approach that is often useful for speeding optimization are multiscale methods and multigrid methods.
**Stochastic MPM Algorithm:**

/* Generate $L$ samples from the posterior distribution */
For $l = 1$ to $L - 1$ {
    Generate a stochastic sample from the distribution
    
    $X^{(l)} \sim \frac{1}{z} \exp \{-u(x)\}$

    where
    
    $u(x) = \sum_{s \in S} l(y_s | x_s) + \beta \sum_{\{s,r\} \in C} \delta(x_r \neq x_s)$

    }/* Compute MPM estimate for each pixel */
For each pixel $s \in S$ {
    Compute the conditional histogram
    
    $h_s(x_s) = \frac{1}{L} \sum_{l=0}^{L-1} \delta(X^{(l)}_s - x_s)$.

    Compute approximate MPM estimate
    
    $\hat{x}_s \leftarrow \arg \max_{x_s} h_s(x_s)$

}
Consider the somewhat more general MAP cost function with the form
\[ u(x) = \sum_{s \in S} l(y_s|x_s) + \beta_1 t_1(x) + \beta_2 t_2(x) , \]
where \( t_1(x) \) and \( t_2(x) \) count the number of horizontal-vertical and diagonal pixel pairs, respectively. This is just the same as using the \( M \)-level discrete MRF of equation (14.17) as a prior model. So our strategy is to first compute the MAP estimate under the assumption that the segmentation is constant on blocks of size \( 2^l \times 2^l \). Then we repeat with smaller blocks until we reach a block size of \( 2^0 = 1 \) at the finest resolution. At each level, we initialize the MAP segmentation process with an interpolated version of the previous lower resolution MAP estimate. Since each MAP optimization is initialized with a good first guess, the convergence is rapid and tends to avoid local minimum. So the entire coarse-to-fine process is typically fast and results in a good quality segmentation.

More specifically, we can adopt a quad-tree structure as shown in Figure 16.6(a). So the finest resolution is denoted by \( l = 0 \) and the coarsest by \( l = L - 1 \). At each resolution, a single pixel represents a block of size \( 2^l \times 2^l \) at the finest scale. If we assume that each entire block has a single class, then the associated negative log likelihood function at scale \( l \) and pixel \( s \) is given by
\[ \ell^{(k)} (m) = \sum_{r \in d(s)} \ell^{(k-1)} (m) , \]
where \( d(s) \) denotes the four children of \( s \), and this fine-to-coarse recursion is initialized with \( \ell^{(0)} (m) = l(y_s|m) \) at the finest scale. In fact, implementing this recursion at all scales is quite efficient requiring only \( \mathcal{O}(N) \) operations.

We can also define corresponding course scale statistics, \( t_1^{(l)} \) and \( t_2^{(l)} \), that count the number of fine scale pixel pairs of differing type that result from this course scale segmentation. If we assume that the segmentation is constant on blocks, as is the case if the segmentation is performance at scale \( l \), then the values of \( t_1^{(l)} \) and \( t_2^{(l)} \) at each scale are related by the following recursion.
\[ t_1^{(l-1)} = 2t_1^{(l)} \tag{16.11} \]
\[ t_2^{(l-1)} = 2t_1^{(l)} + t_2^{(l)} \tag{16.12} \]
To see why this is true, consider Figure 16.6(b). Notice that for each adjacent block of different class at scale \( l \), there are 2 adjacent blocks at scale \( l - 1 \) of
Figure 16.7: A pseudocode specification of the multiresolution segmentation (MRS) algorithm for computing an approximate MAP estimate of a discrete random field $X$. Multiresolution algorithms can often avoid local minima in the MAP cost function and are computationally efficient.

different class, and 2 diagonal blocks at scale $l-1$ of different class. Also, for each diagonal block of different class at scale $l$, there is only 1 diagonal block at scale $l-1$ of different class. Summing terms results in equations (16.11) and (16.12). In matrix form, this can be expressed as

\[
\begin{bmatrix}
t_1^{(l-1)} \\
t_2^{(l-1)}
\end{bmatrix}
= \begin{bmatrix}
2 & 0 \\
2 & 1
\end{bmatrix}
\begin{bmatrix}
t_1^{(l)} \\
t_2^{(l)}
\end{bmatrix}.
\] (16.13)

In order for the optimization problem to be consistent at different scales, the values of the energy functions at scales $l$ and $l-1$ should be equal. Formally, this means that the following equality must hold.

\[
\begin{bmatrix}
\beta_1^{(l-1)} & \beta_1^{(l-1)} \\
\beta_1^{(l-1)} & \beta_1^{(l-1)}
\end{bmatrix}
\begin{bmatrix}
t_1^{(l-1)} \\
t_2^{(l-1)}
\end{bmatrix}
= \begin{bmatrix}
\beta_1^{(l)} & \beta_1^{(l)} \\
\beta_1^{(l)} & \beta_1^{(l)}
\end{bmatrix}
\begin{bmatrix}
t_1^{(l)} \\
t_2^{(l)}
\end{bmatrix}.
\] (16.14)
Then substituting in equation [16.13] into equation [16.14], results a recursion between the parameters, $\beta$, at scale $l - 1$ and $l$.

$$
\begin{bmatrix}
\beta_1^{(l)} \\
\beta_2^{(l)}
\end{bmatrix} =
\begin{bmatrix}
2 & 2 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
\beta_1^{(l-1)} \\
\beta_2^{(l-1)}
\end{bmatrix}.
$$

(16.15)

So the parameters $\beta^{(0)}$ can be chosen at the finest scale, and then the recursion of equation (16.15) can be used to generate parameters $\beta^{(l)}$ at all desired coarser scales. Using these coarser scale parameters, the MAP segmentation problem at scale $l$ then becomes

$$
\hat{x}^{(l)} = \arg \min_{x^{(l)}} \left\{ \sum_s l_s^{(l)}(x_s^{(l)}) + \beta_1^{(l)} t_1^{(l)} + \beta_2^{(l)} t_2^{(l)} \right\}.
$$

Interestingly, this is the same basic problem as we had at the finest scale, but only with different values of the MRF parameters $\beta_1^{(l)}$ and $\beta_2^{(l)}$ and new negative log likelihood functions, $l_s^{(l)}(x_s^{(l)})$.

The pseudocode for this multiresolution segmentation algorithm is given in Figure [16.7]. In practice, the values of $\beta_1^{(l)}$ and $\beta_2^{(l)}$ at different scales can be adjusted empirically. One simple choice is to just leave them constant, which is roughly equivalent to starting with less regularization at course scales and then increasing the regularization with each successive finer scale.

### 16.2 Joint Segmentation and Reconstruction

Reconstruction and segmentation are typically viewed as distinct operations, with reconstruction focused on forming an image with continuously valued pixels, and segmentation focused on assigning each pixel to a specific discrete class. However, model-based imaging provides a powerful framework for the integration of these two seemingly distinct operations [24, 20].

Consider a problem in which the observed data, $Y$, is formed from a linear transformation of some underlying image, $X$; but in addition, we know that each pixel in $X$ takes on one value from a discrete set of $M$ unknown gray levels. We will denote these $M$ unknown gray levels of the image by the

---

4The approach in this section follows the method described in [24].
parameter vector $\phi = [\phi_0, \cdots, \phi_{M-1}]^t$. Then each individual pixel in $X$ must have the form

$$X_s = \phi_{Z_s},$$

where $Z_s \in \{0, \cdots, M - 1\}$ is the discrete class of the $s^{th}$ pixel.

So for example, this is a reasonable model to assume in many real-world tomographic reconstruction problems. In X-ray transmission tomography, the measurements, $Y_s$, are formed from 2 or 3-D integral projections of the image, $X$, along lines. Typically, these measurements are noisy and sometimes incomplete due to physical limitations or opaque obstructions in the object. Since these projections are linear integrals, the model for this problem is exactly of the form of equation (7.4) from Chapter 6. So, we have that

$$Y = AX + W,$$

where $A$ is a sparse matrix with entries $A_{i,j}$ that are proportional the intersection of the $i^{th}$ projection ray with the $j^{th}$ pixel.

In many tomographic imaging applications, it is reasonable to assume the object being imaged comprises a set of discrete materials. So for example, the object might be made of air, plastic, aluminum, etc. In this case, we know that each pixel, $X_s$, takes on only one of $M$ distinct values depending in which material the pixels $s$ lies. So if $\phi = [\phi_0, \cdots, \phi_{M-1}]^t$ represents the $M$ possible X-ray densities corresponding to $M$ materials, then each pixel can be represented as $X_s = \phi_{Z_s}$, where $Z_s$ is the class of the $s^{th}$ pixel. Taking some notational liberties, we may write that $X = \phi Z$, and the forward model of the data given the unknowns, $X$ and $\phi$, may be written as

$$Y = A\phi Z + W. \quad (16.16)$$

This new system model of equation (16.16) creates a powerful synergy between both a continuous and discrete representations of the problem. Intuitively, it can dramatically reduce the amount of data required to solve the associated inverse problem by putting a strong constraint on the solution. Rather than requiring a general solution $X \in \mathbb{R}^N$, we need only estimate the $M$ continuously valued densities together with the discrete classifications of each pixel.

So our objective will be to simultaneously estimate $Z$ and $\phi$. This can be done in many ways, but here we choose to use the joint-MAP estimator of
Section 11. To do this, we first define the MAP cost function given by

\[
f(z, \phi) = -\log p(y|z, \phi) - \log p(z) + \text{const}
\]

\[
= \frac{1}{2}||Y - A\phi_z||_A^2 + \sum_{\{s,r\} \in \mathcal{C}} \beta_{r-s} \delta(z_r \neq z_s),
\]

and then the joint-MAP estimate of equation (11.1) is given by

\[
(\hat{z}, \hat{\phi}) = \arg \min_{z \in \Omega^N, \phi \in \mathbb{R}^M} f(z, \phi),
\]

where \(\Omega = \{0, \cdots, M - 1\}\). In many cases, it is reasonable to enforce the additional constraint of positivity, so that \(\phi \in \mathbb{R}^{+M}\). It is relatively straightforward to do this, but to keep things simple, we will not include a positivity constraint here.

Now in order to make the problem easier to solve, we will introduce some new notation. Let \(J_z\) be an \(N \times M\) matrix with binary entries given by

\[
[J_z]_{i,j} = \delta(j - 1 - z_i),
\]

where the pixels are assumed to be indexed in raster order so that \(i = 1, 2, \cdots, N\). So for example, if \(M = 3\) and \(z = [0, 2, 1, 1]^t\), then

\[
J_z = \begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
0 & 1 & 0
\end{bmatrix},
\]

so that every row of \(J_z\) has a single entry of 1 in the location corresponding to class \(z_s\). With this structure, we can then specify the gray level image \(X\) as

\[
X = J_z \phi,
\]

where the \(s^{th}\) row of the matrix \(J_z\) selects the entry of the vector \(\phi\) corresponding the the class \(z_s\). Using this new notation, we can now express the MAP cost function as

\[
f(z, \phi) = \frac{1}{2}||Y - AJ_z \phi||_A^2 + \sum_{\{s,r\} \in \mathcal{C}} \beta_{r-s} \delta(z_r \neq z_s). \tag{16.17}
\]
Now to minimize this MAP cost function, we will use alternating minimization. More specifically, we need to iteratively solve the following two optimization problems.

\[ z \leftarrow \arg \min_z f(z, \phi) \quad (16.18) \]
\[ \phi \leftarrow \arg \min_\phi f(z, \phi) \quad (16.19) \]

In practice, each alternating update need not exactly minimize the cost function with respect to its variable. Even an update of the variable that only reduces the cost is still very useful and can lead to a local minimum of the cost function.

Equation (16.18) is simply the MAP estimate for the reconstruction of a discrete valued image. So our objective will be to perform an update on the discrete image, \( z \), that reduces the MAP cost function. Since \( z \) is discretely valued, methods such as gradient descent and conjugate gradient cannot be applied because the gradient of \( f(z, \phi) \) with respect to \( z \) does not exist. So instead we will use the method of ICM introduced in Section 16.1.1. However, since ICM is equivalent to ICD, we can also use the methods of Section 5.4 to accelerate the computation of the ICM updates.

In order to derive a simple expression for the update of a pixel, first consider the pixel at location \( s \) with gray level given by \( X'_s = \phi_{z_s} \). If this pixel is changed to be class \( k \), then its updated gray level will be given by \( X_s = \phi_k \). Recalling that \( X = J_z \phi \), then the log likelihood term of the cost function can be written as a quadratic function of the individual pixel’s gray value,

\[ \frac{1}{2} \| Y - AJ_z \phi \|_A^2 = \frac{1}{2} \| Y - A (X + (X_s - X'_s)\varepsilon_s) \|_A^2 = \theta_1 (X_s - X'_s) + \frac{1}{2} \theta_2 (X_s - X'_s)^2 + \text{const} = \theta_1 (\phi_k - \phi_{z_s}) + \frac{1}{2} \theta_2 (\phi_k - \phi_{z_s})^2 + \text{const} \]

where \( \varepsilon_s \) is a vector that is 1 for the element \( s \) and 0 otherwise, and the coefficients \( \theta_1 \) and \( \theta_2 \) are computed in much the same way as they were for

---

5The cost function should be reduced by some percentage of the possible maximum reduction, and this percentage should be bounded above zero. Otherwise, it is possible to have an asymptotically decreasing reduction that will converge to a value that is bounded away from even a local minimum.
Discrete-Continuous ICD Algorithm:

Initialize $z$ and $\phi$

Initialize $e = y - A\phi_z$

For $K$ iterations {
   For each pixel $s \in S$ {
      $\theta_1 \leftarrow -e^t \Lambda A^s_s$
      $\theta_2 \leftarrow A^s_s t \Lambda A^s_s$
      $z_{old} \leftarrow z_s$
      $z_s \leftarrow \arg \min_{k \in \{0, M-1\}} \left\{ \theta_1(\phi_k - \phi_{zs}) + \frac{1}{2} \theta_2(\phi_k - \phi_{zs})^2 + \sum_{r \in \partial s} \beta_{s-r} \delta(z_r \neq k) \right\}$
      $e \leftarrow e + A^s_s(\phi_{s} - z_{old})$
   }
   $R \leftarrow J^t_s A^t \Lambda A J_s$
   $b \leftarrow J^t_s A^t Y$
   $\phi \leftarrow R^{-1} b$
   $e \leftarrow y - A\phi_z$
}

Figure 16.8: A pseudocode specification of the mixed discrete-continuous version of the ICD optimization algorithm for computing an approximate joint-MAP estimate of $(z, \phi)$. The actual reconstruction is then given by $x_s = \phi_{zs}$ where the values $\phi_m$ represent the gray levels associated with each of the $M$ classes, and $z_s$ represents the discrete segmentation of the image.

The ICD update parameters of equations (7.18) and (7.19).

$$\theta_1 = -e^t \Lambda A^s_s$$
$$\theta_2 = A^s_s t \Lambda A^s_s .$$

Using this expression, the ICD update corresponding to equation (7.17) is given by

$$x_s \leftarrow \arg \min_{k \in \{0, M-1\}} \left\{ \theta_1(\phi_k - \phi_{zs}) + \frac{1}{2} \theta_2(\phi_k - \phi_{zs})^2 + \sum_{r \in \partial s} \beta_{s-r} \delta(z_r \neq k) \right\} .$$

Equation (16.19) updates the parameter vector $\phi$ to maximize the MAP.
cost function. With the quadratic form of equation (16.17), the update of $\phi$ is simply the solution to a linear set of equations. More specifically,

$$
\phi \leftarrow \text{arg min}_\phi \frac{1}{2} \phi^T R \phi - b^T \phi = R^{-1} b
$$

where

$$
R = J_z^t A^t \Lambda A J_z
$$

$$
b = J_z^t A^t Y.
$$

16.3 ML Parameter Estimation with Stochastic EM

One huge advantage of model-based methods is that model parameters can be estimated during the inversion process. So for example, in Section 11 we saw that the model for the observations, $Y$, and unknown, $X$, often includes unknown parameters sometimes called hyper-parameters.

More specifically, the joint distribution of $(Y, X)$ can often be written in the form

$$
\log p(y, X|\phi, \theta) = \log p(y|X, \phi) + \log p(X|\theta), \quad (16.20)
$$

where $\phi$ parameterizes the system model, and $\theta$ parameterizes the prior distribution. So for example, $\phi$ might represent unknown parameters of the noise amplitude or perhaps the point-spread function of an imaging system; and $\theta$ can represent the unknown scale parameter of an MRF that controls the level of regularization.

Now, if both $Y$ and $X$ are known, then these parameters can be estimated using standard estimators such as the ML estimate. However, this is usually not the case. In fact, it is almost always the case that $X$ is not known since the objective of Bayesian estimation is to determine $X$!

The EM algorithm introduced in Chapter 12 is precisely designed to handle this missing data problem, so an obvious solution is to simply use it to estimate $(\phi, \theta)$ from $Y$. However, there is a catch. In Section 13.3.3, the EM algorithm was used to estimate the parameters of an HMM; but in this case, we could also use the forward-backward algorithm of Section 13.3.2 to exactly compute the posterior probabilities of $X$ given $Y$. However, when $X$ is a 2-D
MRF, then generally it is not possible to compute the posterior distribution in closed from; so it is therefore, not possible to compute the E-step of the EM algorithm in closed from either. As a partial solution to this dilemma, in Section 11 we introduce the joint-MAP estimate of equations (11.10) and the adjustment factor approach of equation (11.11) as heuristic techniques that do not employ the true ML estimator.

However, if one desires to compute the true ML estimate of parameters \((\phi, \theta)\) it is possible to do this numerically in a tractable way by using the Monte Carlo sampling techniques of Chapter 15 and Section 16.1.2. In order to do this, we generate samples from the posterior distribution using the Hastings-Metropolis algorithm, and then we use those samples to approximate the expectation required for the E-step in the EM algorithm. This approach to the EM algorithm using Monte Carlo sampling is known as stochastic EM. In order to implement it, we must first generate samples from a Gibbs distribution using an energy function with the form of equation (16.11), so that

\[
u(x) = -\log p(y|x) - \log p(x) + \text{const}.
\]

Then, \(L\) samples are generated from the posterior distribution by sampling from the Gibbs distribution

\[
X^{(l)} \sim \frac{1}{z} \exp \{-u(x)\}.
\]

In general, the EM update equations for the estimation of \(\phi\) and \(\theta\) are given by

E-step: \(q(\phi, \theta; \phi^{(k)}, \theta^{(k)}) = E[\log p(y, X|\phi, \theta)|Y = y, \phi^{(k)}, \theta^{(k)}]\)

M-step: \((\hat{\phi}^{(k+1)}, \hat{\theta}^{k+1}) = \arg \max_{\phi, \theta} q(\phi, \theta; \phi^{(k)}, \theta^{(k)})\)

However, using the structure of equation (16.20), we can decompose the \(q\) function into its two components resulting in the decoupled EM updates given
16.3 ML Parameter Estimation with Stochastic EM

below.

\[
E\text{-step:} \quad q_1(\phi; \phi^{(k)}, \theta^{(k)}) = E[\log p(y|X, \phi)|Y = y, \phi^{(k)}, \theta^{(k)}]
\]
\[
E\text{-step:} \quad q_2(\theta; \phi^{(k)}, \theta^{(k)}) = E[\log p(X|\theta)|Y = y, \phi^{(k)}, \theta^{(k)}]
\]

\[
M\text{-step:} \quad \hat{\phi}^{(k+1)} = \arg \max_{\phi} q_1(\phi; \phi^{(k)}, \theta^{(k)})
\]
\[
M\text{-step:} \quad \hat{\theta}^{k+1} = \arg \max_{\theta} q_2(\theta; \phi^{(k)}, \theta^{(k)})
\]

Next we can use the methods introduced in Section 12.7 to simplify the EM expressions when the distributions are exponential, which is almost always the case. More specifically, if we assume that \( T_1(y, x) \) is a natural sufficient statistic for \( \phi \), and \( T_2(x) \) is a natural sufficient statistic for \( \theta \), then the ML estimate of \( \phi \) and \( \theta \) from the complete data, \( (Y, X) \), can be written as

\[
\hat{\phi} = f_1(T_1(Y, X))
\]
\[
\hat{\theta} = f_2(T_2(X))
\]

for some function \( f_1(\cdot) \) and \( f_2(\cdot) \). Then using the methods of Section 12.7.2, we can derive simplified updates for the EM algorithms based on the expectation of the natural sufficient statistics.

\[
E\text{-step:} \quad \bar{T}_1 = E \left[ T_1(y, X)|Y = y, \phi^{(k)}, \theta^{(k)} \right]
\]
\[
E\text{-step:} \quad \bar{T}_2 = E \left[ T_2(X)|Y = y, \phi^{(k)}, \theta^{(k)} \right]
\]

\[
M\text{-step:} \quad \hat{\phi}^{(k+1)} = f_1(\bar{T}_1)
\]
\[
M\text{-step:} \quad \hat{\theta}^{k+1} = f_2(\bar{T}_2)
\]

Now we simply replace the expectations of the E-step with a average over
the Monte Carlo samples from the posterior distribution.

\[
E\text{-step:} \quad \bar{T}_1 = \frac{1}{L} \sum_{l=0}^{L-1} T_1(y, X^{(l)}) \quad (16.21)
\]

\[
\bar{T}_2 = \frac{1}{L} \sum_{l=0}^{L-1} T_2(X^{(l)})
\]

\[
M\text{-step:} \quad \hat{\phi}^{(k+1)} = f_1(\bar{T}_1) \quad (16.22)
\]

\[
\hat{\theta}^{k+1} = f_2(\bar{T}_2)
\]

The following examples shows how this approach can be applied to the specific example of estimating the parameters of both an MRF, \(X\), and the system model, \(Y\) given \(X\).

In the first example, we illustrate how the stochastic EM algorithm can be used to estimate the parameters, \(\theta\), of an unobserved MRF, \(X\). However, to do this, we must be able to compute the ML estimate of \(\theta\) when \(X\) is directly observed. From Section 14.5, we know from that the ML estimates of MRF parameters are, in general, difficult to compute due to the often intractable nature of the partition function.

So in order to simplify the example, we will consider a case for which the ML estimate of the parameter, \(\theta\), has closed form. We note that for a discrete valued MRF, the ML parameter estimate, \(\theta\), typically never has closed form. However, approaches do exist for numerically approximating the ML estimate for discrete MRFs, so these methods can also be used.

**Example 16.1.** Here we provide an example of using stochastic EM to estimate the scale parameter of a continuous MRF prior model from noisy observations of a linear transformation of the image. More specifically, we will assume that that \(X\) has the form of a pair-wise Gibbs distribution using a GGMRF potential function with an unknown scale parameter, \(\sigma_s\).

From Section 6.4, we know that the prior distribution has the form

\[
p(x|\sigma_s) = \frac{1}{z} \exp \left\{ -\frac{1}{p\sigma_s^p} \sum_{\{s,r\} \in P} b_{s,r}|x_s - x_r|^p \right\}, \quad (16.23)
\]
and the ML estimate of $\sigma_x$ is given by equation (16.23) as

$$\hat{\sigma}_x^p = \frac{1}{N} \sum_{\{s,r\} \in P} b_{s,r} |x_s - x_r|^p.$$  \hspace{1cm} (16.24)

So if we assume a model with the form of equation (7.4), then the associated energy function for the Gibbs distribution is exactly the MAP cost function from equation (7.21) and is given by

$$u(x; \sigma_x) = \frac{1}{2} ||y - Ax||_\Lambda^2 + \frac{1}{p\sigma_x^p} \sum_{\{s,r\} \in P} b_{s,r} |x_s - x_r|^p.$$  \hspace{1cm} (16.25)

In fact, it can be shown that the statistic

$$T(x) = \sum_{\{s,r\} \in P} b_{s,r} |x_s - x_r|^p$$  \hspace{1cm} (16.26)

is the natural sufficient statistic for the parameter $\sigma_x$ of the exponential distribution of (16.23), and that the ML estimate of $\sigma_x$ given in equation (16.24) has the form

$$\hat{\sigma}_x = (T(x))^{1/p}.$$  

So putting this all together with the updates of equations (16.21) and (16.22) results in the stochastic EM update steps,

\textbf{E-step:} \hspace{0.5cm} $X^{(l)} \sim \frac{1}{z} \exp \left\{-u(x; \sigma_x^{(k)})\right\}$ for $l = 1$ to $L - 1$

\textbf{M-step:} \hspace{0.5cm} $\hat{\sigma}_x^{(k+1)} = \left\{ \frac{1}{LN} \sum_{l=0}^{L-1} \sum_{\{s,r\} \in P} b_{s,r} |X_s^{(l)} - X_r^{(l)}|^p \right\}^{1/p}$

where again $\theta^{(k)} = \sigma_x^{(k)}$ is the unknown scale parameter for the MRF. In practice, this stochastic EM update can often be performed with only $L = 1$ samples from the posterior. This is because even just a single image will typically have enough pixels to get an accurate estimate of $\sigma_x$.

Our next example is for the estimation of the parameter vector $\phi$ when $X$ is an $M$-level discrete MRF, and the observations, $Y$, are conditionally
Gaussian with unknown means and variances. In this case, the samples from the posterior distribution can serve double duty by both being used to perform the E-step of stochastic EM algorithm, while also being used to compute the MPM estimate of $X$ as described in Section 16.1.2. The resulting algorithm is known as the EM/MPM algorithm of [17].

**EM/MPM Algorithm:**

```plaintext
/* Compute EM updates */
Repeat until converged {
    E-step: For $l = 1$ to $L - 1$
        $X^{(l)} \sim \frac{1}{Z} \exp \{-u(x; \phi^{(l)})\}$
        
    M-step:
        $\hat{N}_{m}^{(k+1)} = \frac{1}{L} \sum_{s \in S} \sum_{l=0}^{L-1} \delta(X^{(l)} - m)$
        $\hat{\mu}_{m}^{(k+1)} = \frac{1}{\hat{N}_{m}} \sum_{s \in S} \sum_{l=0}^{L-1} Y_{s} \delta(X^{(l)} - m)$
        $\hat{\sigma}_{m}^{2(k+1)} = \frac{1}{\hat{N}_{m}} \sum_{s \in S} (Y_{s} - \hat{\mu}_{m})^{2} \delta(X^{(l)} - m)$
}

/* Compute MPM estimate */
For each pixel {
    Compute the conditional histogram
    $h_{s}(x_{s}) = \frac{1}{L} \sum_{l=0}^{L-1} \delta(X_{s}^{(l)} - x_{s})$.

    Compute approximate MPM estimate
    $\hat{x}_{s} \leftarrow \arg \max_{x_{s}} h_{s}(x_{s})$
}
```

Figure 16.9: A pseudocode specification of the EM/MPM algorithm. The stochastic sampling methods of Chapter 15 are used to generate samples from the posterior distribution. Then these samples are used both to perform the E-step of the EM algorithm, and also to compute the MPM estimate of the segmentation $X$.

**Example 16.2.** In this example, we consider the case when our objective is to segment an image by estimating the discrete 2-D MRF, $X$, from obser-
observations, $Y$, and we must simultaneously estimate the mean and variance for the observations of each class.

Assume that each pixel, $Y_s$, is conditionally Gaussian with distribution $N(\mu_{x_s}, \sigma^2_{x_s})$, but the particular values of the parameters $\mu_m$ and $\sigma^2_m$ are unknown for the $M$ classes, $m \in \{0, \cdots, M-1\}$. So then the negative log likelihood is given by

$$l(y_s|x_s, \phi) = \frac{1}{2\sigma^2_{x_s}} (y_s - \mu_{x_s})^2 + \frac{1}{2} \log \left(2\pi \sigma^2_{x_s}\right),$$

and the MAP cost function is given by

$$u(x; \phi) = \sum_{s \in S} l(y_s|x_s, \phi) + \beta \sum_{\{s,r\} \in \mathcal{C}} \delta(x_r \neq x_s),$$

where $\phi = \{\mu_0, \sigma^2_0, \cdots, \mu_{M-1}, \sigma^2_{M-1}\}$ is the unknown parameter that we would like to estimate.

Now if both $Y$ and $X$ were available, then the ML estimates of $\mu_m$ and $\sigma^2_m$ can be easily estimated as

$$N_m = \sum_{s \in S} \delta(X_s - m)$$
$$\hat{\mu}_m = \frac{1}{N_m} \sum_{s \in S} Y_s \delta(X_s - m)$$
$$\hat{\sigma}^2_m = \frac{1}{N_m} \sum_{s \in S} (Y_s - \hat{\mu}_s)^2 \delta(X_s - m).$$

However, if the $X$ is not known, then we can use the EM algorithm to estimate the parameters $\mu_m$ and $\sigma^2_m$. To do this, we simply take the posterior expectation of the natural sufficient statistics as described in more detail in Section 12.7.2. This results in the following EM-update.

$$\hat{N}_{m}^{(k+1)} = \sum_{s \in S} P\{X_s = m|Y = y, \phi^{(k)}\}$$
$$\hat{\mu}^{(k+1)}_m = \frac{1}{\hat{N}_m} \sum_{s \in S} Y_s P\{X_s = m|Y = y, \phi^{(k)}\}$$
$$\hat{\sigma}^2_{m}^{(k+1)} = \frac{1}{\hat{N}_m} \sum_{s \in S} (Y_s - \hat{\mu}_s)^2 P\{X_s = m|Y = y, \phi^{(k)}\}$$
Of course, the problem with this approach is that there is no closed form expression for the probability \( P\{X_s = m|Y = y, \phi^{(k)}\} \) when \( X \) has a discrete multi-level MRF distribution. So instead we generate \( L \) samples from the posterior distribution \( X^{(l)} \sim \frac{1}{z} \exp \{-u(x; \phi)\} \), and then we use these samples to compute the approximate posterior probabilities.

This results in the following stochastic EM update steps,

**E-step:** \( X^{(l)} \sim \frac{1}{z} \exp \{-u(x; \phi^{(k)})\} \)

**M-step:**

\[
\hat{N}_{m}^{(k+1)} = \frac{1}{L} \sum_{s \in S} \sum_{l=0}^{L-1} \delta(X^{(l)}_s - m)
\]

\[
\hat{\mu}_m^{(k+1)} = \frac{1}{L\hat{N}_m} \sum_{s \in S} \sum_{l=0}^{L-1} Y_s \delta(X^{(l)}_s - m)
\]

\[
\hat{\sigma}^2_m^{(k+1)} = \frac{1}{L\hat{N}_m} \sum_{s \in S} \sum_{l=0}^{L-1} (Y_s - \hat{\mu}_m)^2 \delta(X^{(l)}_s - m)
\]

where again \( \phi^{(k)} = \{\mu_0^{(k)}, \sigma_0^{2(k)}, \cdots, \mu_{M-1}^{(k)}, \sigma_{M-1}^{2(k)}\} \) are the mean and variance for each classes conditional distribution.
16.4 Chapter Problems

1. Let $f : \mathbb{R}^N \rightarrow \mathbb{R}$ be a continuous convex function which is differentiable everywhere and has a unique global minima. Let $x \in \{0, 1\}^N$ be a binary valued vector with $N$ components. We would like to compute

$$\hat{x} = \arg \min_{x \in \{0, 1\}^N} f(x).$$

a) Is the minimum unique? Prove your answer or give a counter example.

b) Does the Gauss-Seidel/Coordinate search method converge to a global minimum? Prove your answer or give a counter example.

2. Show that $T(x)$ of equation (16.26) is the natural sufficient statistic for the parameter $\sigma_x$ of the exponential distribution of (16.23).
Chapter 17

Poisson Data Models

In the previous chapters, we have modeled the signal as a continuously valued random quantity. However, physics tells us that ultimately physical phenomena come in discrete quanta. In electromagnetic imaging systems, such as optical cameras or X-ray scanners, these discrete quanta correspond to individual photons of light. In other imaging systems, these discrete events may correspond to the count of detected electrons in an electron microscope, or the count of neutrons in a neutron imaging system. In all these cases, the most sensitive imaging systems measure discrete counts of events rather than continuous signals. In order to optimally process these discrete measurements, we will need to accurately model them; and typically the model of choice is the Poisson distribution.

In this chapter, we will delve into the details of model-based imaging with Poisson data. The detailed mechanics of modeling Poisson data are quite different than in the additive Gaussian noise case. First, the variance of Poisson measurements increases with the mean. This means that the noise actually increases with the signal strength, leading to some counter-intuitive results. Also, Poisson measurements have a uniquely distinct interpretation when the event counts are zero. The observation of “nothing” provides important information in an event counting system, which is both subtly and accurately quantified by the Poisson distribution. However, working with Poisson distributions can be challenging because the associated log likelihood terms, while convex, do not have a quadratic form. In order to address this challenge, we will introduce quadratic surrogate functions that can be used to successively approximate the true Poisson expressions; thus leading to computationally tractable algorithms with guaranteed convergence.
17.1 The Poisson Forward Model

In order to build a foundation for the modeling of event-counting systems, we start by considering the attributes of **Poisson random variables**. Let $Y \in \mathbb{Z}^+$ denote a Poisson random variable. Then its probability mass function (PMF) is given by

$$ p_\lambda(y) \triangleq P_\lambda\{Y = y\} = \frac{\lambda^y e^{-\lambda}}{y!} ,$$

(17.1)

where $\lambda \geq 0$ parameterizes the distribution. For notational simplicity, we write that $Y \sim \text{Pois}(\lambda)$. From this form, it is easily shown that the PMF sums to 1 and that $Y$ has a mean and variance of $\lambda$,

$$ \sum_{y=0}^{\infty} p_\lambda(y) = \sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} = 1 $$

(17.2)

$$ \mathbb{E}[Y] = \sum_{k=0}^{\infty} k \frac{\lambda^k e^{-\lambda}}{k!} = \lambda $$

(17.3)

$$ \mathbb{E}[(Y - \lambda)^2] = \sum_{k=0}^{\infty} (k - \lambda)^2 \frac{\lambda^k e^{-\lambda}}{k!} = \lambda . $$

(17.4)

Intuitively, $\lambda$ represents the average number of detected events that occur in a measurement period; so in many applications, we can also think of it as an event rate when it is normalized by the detection time. Two important special cases occur when $y = 0$ and when $\lambda = 0$. When $\lambda = 0$, the rate is zero, and $p_0(0) = P_0\{Y = 0\} = 1$. So in this case, the only possible observation is that $Y = 0$, and the probability of any other count is 0.\footnote{The form of the PMF in equation (17.1) works in this case, but requires the interpretations that $0^0 = 1$ and $0! = 1$.} When $y = 0$ then zero events have been counted. Of course, this can occur whenever $\lambda \geq 0$. In this case, $p_\lambda(0) = P_\lambda\{Y = 0\} = e^{-\lambda}$ results in an exponentially decaying likelihood with increasing $\lambda$.

For a Poisson model, the negative log likelihood can be written as

$$ l(\lambda) = -\log p(y|\lambda) $$

$$ = \lambda - y \log \lambda + \log (y!) , $$

(17.5)
where we suppress the dependence on the data, $y$, for notational clarity. From this, we can also derive the ML estimate of the parameter $\lambda$ to be

$$\hat{\lambda} = \arg \min_{\lambda \geq 0} l(\lambda) = \arg \min_{\lambda \geq 0} \{ \lambda - y \log \lambda + \log (y!) \} = y .$$

(17.6)

So the ML estimate of the rate is simply the observed number of events, $Y$.

In the Bayesian framework, we will often assume that the rate of the Poisson random variable is itself random. So in this case, the conditional distribution of $Y \in \mathbb{Z}^+$ given $X \in \mathbb{R}^+$ has the form

$$p(y|x) = \frac{x^y e^{-x}}{y!} ,$$

and for notational convenience we write that $Y|X \sim \text{Pois}(X)$ to indicate that the conditional distribution of $Y$ given $X$ is Poisson with rate $X$. When computing Bayesian estimates, we will often want to use the associated log likelihood of equation (17.5) as our forward model for a photon-counting system. In this case, the negative log likelihood and its derivative are given by the following equations.

$$l(x) = x - y \log x + \log (y! ) \quad (17.7)$$

$$l'(x) = 1 - y / x \quad (17.8)$$

Both these functions are plotted in Figure 17.1 for the case of $y = 1$. Notice, that both functions are unbounded at $x = 0$, which is very important and will cause many headaches later. In addition, when $y \neq 0$ the negative log likelihood function is strictly convex while its derivative is strictly concave.

The convexity of the negative log likelihood will be of critical importance in computing MAP estimators. This is because we know from Chapter 7 the strict convexity of the MAP cost function can be used to ensure that the global minimum exists, is unique, and can be efficiently computed. Perhaps surprisingly, concavity of the negative log likelihood’s derivative will also serve a crucial role in the design of convergent optimization algorithms for MAP reconstruction. In this case, the concavity can be used to construct surrogate functions to the log likelihood in a manner similar to that used for the potential functions in Section 8.3 on page 155.

\footnote{When $y = 0$, the functions are linear, so they are both convex and concave.}
17.1.1 Poisson Noise Approximations

While the negative log likelihood function is convex, its form can be more difficult to deal with than the quadratic functions of previous chapters. One simplification is to approximate $l(x)$ using a Taylor series about some point of approximation $x'$.

\[
l(x) = x - y \log x + \log (y!)
\]

\[
\approx a + b(x - x') + \frac{c}{2}(x - x')^2
\]

where the Taylor series coefficients are given by

\[
a = x' - y \log(x') + \log(y!),
\]

\[
b = 1 - y/x',
\]

\[
c = y/(x')^2.
\]

One particular case of importance is when the point of approximation, $x'$, is taken to be the ML estimate $x' = y$. In this case, $b = 0$ and $c = 1/y$, so the Taylor series approximation to the negative log likelihood has the form

\[
l(x) \approx \frac{1}{2y} (x - y)^2 + \text{const},
\]
where the term of const = $y - y \log y + \log(y!)$ can usually be dropped since
it is not a function of $x$. The approximation of equation (17.11) is sometimes
referred to as the “Gaussian approximation” to the Poisson distribution. Intu-
tively, this comes from the fact that for large counts the random variable $Y$
can be approximated as Gaussian with mean $\mu = x$ and variance $\sigma^2 = x \approx y$.
However, this is really a misnomer since it is not the probability density that
is being approximated as Gaussian. Instead, it is the log likelihood function
that is being approximated as being quadratic.

The approximation of equation (17.11) is very useful at high count rates
since it converts the problem of MAP estimation with Poisson noise to the
cimpler problem of MAP estimation with Gaussian noise. So for example,
if a vector of measurements, $Y$, are independent Poisson random variables,
then following the approach of [59] the negative log likelihood function used
in the MAP estimate can be approximated as

$$l(x) = \frac{1}{2} \|y - x\|_A^2 + \text{const} \quad (17.12)$$

where

$$A = \text{diag}\{1/y_1, \cdots, 1/y_N\}. \quad (17.13)$$

So this results in what appears to be a Gaussian approximation for the data,
$Y$, except that the data is used to also compute the precision matrix $A$. In
fact, Section 11.2 discusses the problem of parameter estimation for a similar
model that arises out of photon counting data, and presents a method for
adaptively estimating an unknown noise gain factor.

However, approximation of equation (17.12) has two disadvantages. First,
it runs into problems when the photon counts are zero because of the need
to compute the terms $1/y_i$ as the elements of $A$. Second, this quadratic ap-
proximation is only accurate when the event count is large, i.e., $Y_i \gg 1$.
In many applications, this is an excellent approximation; but in some appli-
cations, when the photon counts are very low, it may happen that $Y_i \approx 0$,
and the “Gaussian approximation”, or equivalently the quadratic likelihood
approximation, may be very poor.

An alternative approach to simplification of the Poisson log likelihood is
to use a surrogate function as described in Chapter 8. In Section 17.3 we will
discuss the issue of constructing surrogate functions for Poisson distributions
Figure 17.2: This diagram illustrates a typical scenario for an X-ray imaging system with a photon counting detector. X-ray photons are emitted from some material. A particular voxel may emit photons at a rate $X_j$. However, only a fraction of these photons, $A_{i,j}X_j$, are detected by the $i^{th}$ photon counting detector because a pin-hole collimator blocks all photons that do not fall along the line of interest.

In detail. Quadratic surrogates simplify the update computations while still guaranteeing convergence to the true MAP estimate.

17.1.2 Poisson Models for Emission Tomography

In real applications, it is often the case that each observed Poisson random variable is the sum of photons emitted from more than one underlying source location. Figure 17.2 illustrates a typical scenario in which the $j^{th}$ photon counting detector of an array measures the X-ray emissions along a line through the object. In concept, this model applies to a wide range of imaging applications such as single photon emission tomography (SPECT) and positron emission tomography (PET). In SPECT imaging, a lead collimator is used to form a pin-hole camera by blocking all the photons except those that are emitted along a line.

PET scanners achieve a similar result using a very different principal. The subject in a PET scanner is first injected with a radio-pharmaceutical such as fluorodeoxyglucose (FDG) containing a radioactive isotope, in this case $\text{F}^{18}$, that emits positrons. When the positron collides with an nearby electron, they annihilate and emit a pair of photons in approximately opposite directions. The PET scanner then detects these coincident pairs of photons in opposing pairs of detectors. Each such coincident event is assumed to
have happened along the line connecting the detector pair. So in this way, the events are grouped into virtual line integrals between detectors.

With this setup in mind, let $Z_{i,j}$ denote the number of photons detected by the $i^{th}$ detector that were emitted by the $j^{th}$ voxel. The term voxel refers to a volume element in 3-D and is analogous to a pixel (i.e., picture element) of an image. In this case, $Z_{i,j} \mid X_j \sim \text{Pois}(A_{i,j}X_j)$ where $A_{i,j} \geq 0$ is the probability that a photon emitted from voxel $j$ will be detected by detector $i$, and $X_j$ is the expected number of emissions from voxel $j$. Since the photon counts are all conditionally independent given $X$, we can then write down both the PMF and log likelihood for the entire array of photon counts, $Z$.

$$p_{Z \mid X}(z \mid x) = \prod_{i=1}^{M} \prod_{j=1}^{N} \frac{(A_{i,j}x_j)^{z_{i,j}}e^{-A_{i,j}x_j}}{z_{i,j}!}$$

$$\log p_{Z \mid X}(z \mid x) = \sum_{i=1}^{M} \sum_{j=1}^{N} \{-A_{i,j}x_j + z_{i,j} \log (A_{i,j}x_j) - \log(z_{i,j}!)}$$

From this it is also clear that the conditional expectation of $Z_{i,j}$ is given by

$$\mathbb{E}[Z_{i,j} \mid X] = A_{i,j}X_j$$

However, typically the detector will not be able to distinguish between different photons; so in this case, the measured photon count will correspond to the sum of all photons detected along the line.

$$Y_i = \sum_{j=1}^{N} Z_{i,j}.$$ 

It is easily shown that the sum of independent Poisson random variables is also a Poisson random variable; so therefore we know that

$$Y_i \mid X \sim \text{Pois}(A_{i,*}X),$$

where $A_{i,*}$ denotes the $i^{th}$ row of the matrix $A$. Furthermore, since each $Y_i$ is formed by the sum of distinct and independent counts, $Z_{i,j}$, the individual measurements $Y_i$ must all be independent. Therefore, we may compactly express the distribution of the measured vector as $Y \mid X \sim \text{Pois}(AX)$. From this, we can write the log likelihood of the measured count vector, $Y$, given
\[ \log p(y|x) = \sum_{i=1}^{M} \left\{-A_{i,*}x + y_i \log (A_{i,*}x) - \log(y_i!}\right\}, \quad (17.17) \]

and the associated negative log likelihood function.

\[ l(x) = \sum_{i=1}^{M} \left\{A_{i,*}x - y_i \log (A_{i,*}x) + \log(y_i!)\right\} \quad (17.18) \]

In practice, one might want to use this forward model to compute the MAP estimate of \( X \). In this case, the MAP estimate would then have the form

\[ \hat{x} = \arg \min_{x \in \mathbb{R}^+ N} \{l(x) + u(x)\}, \quad (17.19) \]

where \( u(x) = -\log p(x) \) is the stabilizing function associated with the prior model and \( x \in \mathbb{R}^+ N \) enforces the positivity constraint. Notice that the positivity constraint along with the fact that the elements of \( A \) are non-negative ensure that the Poisson rates, \( Ax \), are also non-negative.

Moreover, if \( u(x) \) is a convex function, then the optimization problem of equation (17.19) requires the minimization of a strictly convex function on a convex set; so the results of Appendix A apply and it is generally possible to compute a unique global minimum to the MAP optimization problem.

### 17.2 Poisson Surrogate Functions for MAP Estimation

While the Poisson negative log likelihood is convex, it doesn’t have a simple quadratic form that is easy to minimize in MAP estimation. One approach to simplification of the Poisson log likelihood is to use a surrogate function as described in Chapter 8. Surrogate functions work by approximating a function with an upper bound that is typically (but not always) quadratic. Surrogates allow one to have ones cake and eat it too by making computations simple and still guaranteeing convergence to the true minimum.

The following two sections describe methods for efficiently computing Poisson surrogate function. Section [17.3] treats the case of quadratic surrogate functions used for ICD optimization; and Section [17.4] explains out the EM algorithm can be used to generate surrogate Poisson functions.
However, using surrogate functions with Poisson data is tricky because the negative log likelihood function plotted in Figure 17.1(a) is unbounded on the interval $x \in [0, \infty)$. Therefore, no quadratic function can bound $l(x)$ for $x \geq 0$. Our solution to this problem will be to allow estimates to get arbitrarily close to zero, but not actually ever reach zero. In particular, Theorem [17.3.1] below provides a key piece of machinery for constructing surrogate functions to Poisson log likelihoods using the methods similar to those in [73, 74].

**Theorem 17.3.1. Optimal Surrogates for convex functions with concave derivatives**

Let $f : [x_{\text{min}}, \infty) \to \mathbb{R}$ be a convex function with derivative $f' : [x_{\text{min}}, \infty) \to \mathbb{R}$ that is concave. Then for all $x \in [x_{\text{min}}, \infty)$ and $x' \in (x_{\text{min}}, \infty)$ define

$$Q(x; x') = b(x - x') + \frac{c}{2}(x - x')^2$$

where $b$ and $c$ are given by

$$b = f'(x')$$
$$c = 2\frac{f(x_{\text{min}}) - f(x') + f'(x')(x' - x_{\text{min}})}{(x' - x_{\text{min}})^2}$$

is a surrogate function for minimization of $f(x)$ on $x \in [x_{\text{min}}, \infty)$. Furthermore, $Q$ provides the tightest upper bound among all quadratic surrogate functions on $[x_{\text{min}}, \infty)$.

**Proof.** See Appendix C.

In order to better understand the intuition behind Theorem [17.3.1] consider the surrogate function

$$q(x; x') = Q(x; x') + f(x') .$$

Then $q(x; x')$ possesses the two essential properties of an upper bounding surrogate function.

$$f(x') = q(x'; x')$$
$$f(x) \leq q(x; x')$$
In Theorem 17.3.1, the selection of \( b \) ensures that \( \frac{\partial q(x; x')}{\partial x} \bigg|_{x=x'} = f'(x') \), so that the derivative of the surrogate matches the derivative of the function at the point of approximation. In addition, it can be easily shown that the selection of the coefficient \( c \) as specified in equation (17.21) ensures that \( f(x_{\text{min}}) = q(x_{\text{min}}; x') \), so that the surrogate exactly matches the function at the lower bound of the interval.

In order to get more intuition as to how this works, we can apply this property to the Poisson log likelihood given in equation (17.7). So if we choose \( x' \) as our point of approximation, then our search interval will be \( x \in [x_{\text{min}}, \infty) \) where \( 0 < x_{\text{min}} < x' \). In this case the surrogate function is given by

\[
l(x; x') = b(x - x') + \frac{c}{2}(x - x')^2 \tag{17.22}
\]

where

\[
b = 1 - y/x' \tag{17.23}
\]

\[
c = 2 \frac{(b - 1)(x' - x_{\text{min}}) + y \log (x'/x_{\text{min}})}{(x' - x_{\text{min}})^2}. \tag{17.24}
\]

Figure 17.3(a) and (b) illustrates the result for \( x' = 2 \) and two values of \( x_{\text{min}} \). Both sub-figures show a negative log likelihood function, \( l(x) \), for \( y = 1 \); so in both cases, the function takes on its minimum at \( x = y = 1 \). However, in Figure 17.3(a) the minimum allowed value is taken to be \( x_{\text{min}} = 0.2 \) and in Figure 17.3(b) it is taken to be \( x_{\text{min}} = 0.5 \). Notice that the larger value of \( x_{\text{min}} = 0.5 \) results in a tighter bound; however, this is at the expense of a smaller interval of \([0.5, \infty)\). So in this case, the surrogate function is only valid for \( x \geq 0.5 \). When the interval is enlarged to \([0.2, \infty)\), then the surrogate function bound becomes looser. Generally speaking, a tighter surrogate provides a better approximation, so each update is more aggressive, and convergence is likely to be faster.

So while the surrogate function provides an upper bound, one problem is that it requires the selection of a value \( x_{\text{min}} \). One solution to this problem is to make the choice of this minimum value adaptive. To do this, we will define a user selectable parameter, \( 0 < \gamma < 1 \) and then bound the solution adaptively by setting \( x_{\text{min}} = \gamma x' \). So for example if \( \gamma = 1/10 \), then with each iteration of the surrogate function we can reduce the error by 90%.
17.3 ICD Optimization with Surrogate Functions

Figure 17.3: Plots showing the negative log likelihood, $l(x)$, (blue) and the optimal quadratic surrogate function, $l(x; x')$, (green) for two different values of $x_{min}$. In both cases, $y = 1$ and $x' = 2$, but in (a) $x_{min} = 0.2$ and in (b) $x_{min} = 0.5$. Notice that the surrogate function upper bounds $l(x)$ for $x \in [x_{min}, \infty)$. When $x_{min} = 0.2$, the interval $[x_{min}, \infty)$ is larger, but the bound is looser. Alternatively, when $x_{min} = 0.5$, the interval $[x_{min}, \infty)$ is smaller, but the bound is tighter.

The following example below illustrates how this technique for constructing surrogate functions to the Poisson log likelihood can be used to perform ICD optimization for the computation of the MAP estimate.

**Example 17.1.** Consider the problem when our objective is to compute the MAP estimate of equation (17.19) when $Y|X \sim \text{Pois}(X)$. In this case, the negative log likelihood is given by

$$l(x) = \sum_{i=1}^{M} \left\{ x_i - y_i \log(x_i) + \log(y_i!) \right\}.$$  \hspace{1cm} (17.25)

In order to keep the approach general, we will assume that the prior term, $u(x)$, can be efficiently minimized with respect to individual pixels. More specifically, we will assume that there is a set of functions $u_i(x_i, x_{\partial i})$ that only depend on neighboring pixels so that

$$\arg \min_{x_i} u(x) = \arg \min_{x_i} u_i(x_i, x_{\partial i}).$$
ICD for Poisson Measurements $Y|X \sim \text{Pois}(X)$:

Initialize $0 < \gamma < 1$
Initialize $x > 0$
For $K$ iterations {
    For each pixel $j \in 1, \cdots, N$ {
        $x_{\text{min}} \leftarrow \gamma x_j$
        $b \leftarrow 1 - y_j / x_j$
        $c \leftarrow 2 \frac{(b - 1)x_j(1 - \gamma) - y_j \log(\gamma)}{x_j^2(1 - \gamma)^2}$
        $x_j \leftarrow \arg \min_{\alpha \geq x_{\text{min}}} \left\{ b(\alpha - x_j) + \frac{c}{2}(\alpha - x_j)^2 + u_j(\alpha, x_{\partial j}) \right\}$
    }
}

Figure 17.4: Pseudocode for ICD computation of the MAP estimate with a Poisson forward model of the form $Y|X \sim \text{Pois}(X)$ and a general prior model. Each ICD update assumes the use of a quadratic surrogate function for the Poisson log likelihood. The parameter $0 < \gamma < 1$ determines the minimum allowed value of the pixel update.

So for example if

$$u(x) = \sum_{\{s,r\} \in \mathcal{P}} \rho(x_s - x_r)$$

from equation (6.6), then

$$u_i(x_i, x_{\partial i}) = \sum_{r \in \partial i} \rho(x_i - x_r) .$$

From this, we can write the ICD update for the $i^{th}$ pixel as

$$x_i \leftarrow \arg \min_{\alpha \geq 0} \left\{ l_i(\alpha) + u_i(\alpha, x_{\partial i}) \right\} .$$

where we define the function

$$l_i(\alpha) = \alpha - y_i \log(\alpha) + \log(y_i!) .$$

From this, we can then define a quadratic surrogate function with the form

$$l_i(\alpha; x') = b(\alpha - x') + \frac{c}{2}(\alpha - x')^2 \quad (17.26)$$
where \( x' \) is the initial value of the pixel. So the ICD update then becomes

\[
x_i \leftarrow \arg \min_{\alpha \geq 0} \{ l_i(\alpha; x') + u_i(\alpha, x_{\partial i}) \}.
\]

Now since the function \( l_i(\alpha) \) is unbounded as \( \alpha \to 0 \), we must restrict the search interval to \( \alpha \geq x_{\min} > 0 \). In fact, we can make this minimum adaptive by selecting \( x_{\min} = \gamma x' \) for some fixed value of \( 0 < \gamma < 1 \). Typically, one might select \( 1/10 \geq \gamma \leq 1/4 \). Substituting this value of \( x_{\min} = \gamma x' \) into equation (17.23) and (17.24) then results in the coefficients \( b \) and \( c \) given by

\[
b = 1 - y/x' \tag{17.27}
\]

\[
c = \frac{2(b - 1)x'(1 - \gamma) - y \log(\gamma)}{(x')^2(1 - \gamma)^2} \tag{17.28}
\]

Putting this all together results in the ICD update algorithm given in Figure 17.4.

The surrogate function of Theorem 17.3.1 provides the tightest possible quadratic surrogate function on the interval \([x_{\min}, \infty] \). However, computation of the coefficient \( c \) in equation (17.21) can sometimes be difficult because it requires the evaluation of both \( f(x) \) and its derivative \( f'(x) \). The following property often provides a simpler surrogate function that is not quite so tight, but only requires two evaluations of the derivative, \( f'(x) \). In practice, this is often simpler.

**Theorem 17.3.2.** Simplified surrogates for convex functions with concave derivatives

Let \( f(x) \) be a convex function defined on the interval \( x \in [x_{\min}, \infty) \) with a concave derivative \( f'(x) \). Then the function

\[
Q(x; x') = b(x - x') + \frac{c}{2}(x - x')^2
\]

where \( b \) and \( c \) are given by

\[
b = f'(x') \tag{17.29}
\]

\[
c = \frac{f'(x') - f'(x_{\min})}{x' - x_{\min}} \tag{17.30}
\]

is a surrogate function for minimization of \( f(x) \) on \( x \in [x_{\min}, \infty) \).
The following example shows how the simplified surrogate function can be used for computing the MAP estimate with Poisson observations.

Example 17.2. In this example, we will again consider the problem of Example 17.2, but we will use the simplified quadratic surrogate function of Theorem 17.3.2.

As in the previous example, each ICD pixel update is given by

\[ x_i \leftarrow \arg \min_{\alpha \geq 0} \{ l_i(\alpha; x') + u_i(\alpha, x_{\partial i}) \} \]

where new \( l_i(\alpha; x') \) is the simplified quadratic surrogate again with the form

\[ l_i(\alpha; x') = b(\alpha - x') + \frac{c}{2}(\alpha - x')^2 \]

Once again, we will adaptively choose the minimum value so that \( x_{\text{min}} = \gamma x' \).

We next compute the constants \( b \) and \( c \) using equations (17.29) and (17.30).

This results in

\[ b = 1 - y/x' \]

\[ c = \frac{y}{\gamma x_i^2} \]
Simplified ICD for Poisson Measurements $Y|X \sim \text{Pois}(X)$:

- Initialize $0 < \gamma < 1$
- Initialize $x > 0$
- For $K$ iterations {
  - For each pixel $j \in 1, \ldots, N$ {
    - $x_{\text{min}} \leftarrow \gamma x_j$
    - $b \leftarrow 1 - y_j / x_j$
    - $c \leftarrow \frac{y_j}{\gamma x_j^2}$
    - $x_i \leftarrow \arg \min_{\alpha \geq x_{\text{min}}} \left\{ b(\alpha - x_j) + \frac{c}{2}(\alpha - x_j)^2 + u_j(\alpha, x_{\partial j}) \right\}$
  }

Figure 17.6: Pseudocode for Simplified ICD computation of the MAP estimate with a Poisson forward model of the form $Y|X \sim \text{Pois}(X)$ and a general prior model. This algorithm differs from the one in Figure 17.4 because it uses a simplified surrogate function that results in a simpler calculation for the coefficient $c$.

Putting this all together results in the ICD update algorithm given in Figure 17.6.

This same surrogate function approach can also be applied for the direct minimization of the MAP cost function when $Y|X \sim \text{Pois}(AX)$. In this case, $Y$ is formed from conditionally independent Poisson random variables with a conditional mean of $AX$. In this case, the negative log likelihood is given by

$$l(x) = \sum_{i=1}^{M} \left\{ A_{i,*} x - y_i \log(A_{i,*} x) + \log(y_i!) \right\} . \quad (17.33)$$

Since our objective will be to use ICD optimization, we must consider how the likelihood changes when we update the $j^{th}$ pixel, $x_j$. Referring back to Section 5.4, we can express the update of a single pixel as

$$x \leftarrow x' + \alpha \varepsilon_j , \quad (17.34)$$

where $\varepsilon_j$ is a vector that is 1 for element $j$ and 0 otherwise, $x'$ is the initial image before the pixel is updated, and $x$ is the resulting image after updating.
the $j^{th}$ pixel to take on the value $\alpha$. Substituting equation (17.34) into (17.33) then results in

$$l(x' + \alpha \varepsilon_j) = \sum_{i=1}^{M} \{\bar{y}_i + A_{i,j} \alpha - y_i \log (\bar{y}_i + A_{i,j} \alpha) + \log(y_i !)\} \ .$$

where $\bar{y} = Ax'$. Intuitively, $\bar{y}$ represents the expected photon count for each measurement assuming the initial value of the image. Now dropping any constants, we obtain $l_j(\alpha; \bar{y})$, the negative log likelihood function for the $j^{th}$ pixel, and its derivative $l'_j(\alpha; \bar{y})$.

$$l_j(\alpha; \bar{y}) = \sum_{i=1}^{M} \{A_{i,j} \alpha - y_i \log (\bar{y}_i + A_{i,j} \alpha)\} \quad (17.35)$$

$$l'_j(\alpha; \bar{y}) = \sum_{i=1}^{M} \left(1 - \frac{y_i}{\bar{y}_i + A_{i,j} \alpha}\right) A_{i,j} \quad (17.36)$$

From this, we can once again define a quadratic surrogate function with the form

$$Q_j(\alpha; \bar{y}) = b\alpha + \frac{c}{2} \alpha^2 . \quad (17.37)$$

Now at this point, we can calculate the constants $b$ and $c$ based on either the optimal surrogate function of Theorem 17.1 or the simplified surrogate function of Theorem 17.3.2. Both surrogate functions will use the same expression for $b$, but they will result in different expressions for $c$. In each case, $Q_j(\alpha; \bar{y})$ will be a surrogate for the minimization of $l_j(\alpha; \bar{y})$ on an interval $\alpha \in [\alpha_{min}, \infty)$. Notice that since $\alpha$ represents the change in a pixel value, it can be negative. However, the sum of $\alpha$ and the original pixel value $x'_j$ must be bounded above zero; so we have that $x_j + \alpha \geq x_{min} = \gamma x'_j$. This results in the relationship that

$$\alpha_{min} = (\gamma - 1) x'_j$$

We can calculate an expression for the coefficient $b$ from equation (17.20) by simply differentiating equation (17.37) above and evaluating the result at $\alpha = 0$ to yield

$$b = l'_j(0; \bar{y}) \quad (17.38)$$

$$= \sum_{i=1}^{M} \left(1 - \frac{y_i}{\bar{y}_i}\right) A_{i,j} \quad (17.39)$$
where again $\bar{y} = Ax'$. The expression for $c$ is calculated using equation (17.21) using the likelihood function $l_i(\alpha; \bar{y})$ in place of $f(x)$. This results in

$$c = 2 \frac{\Delta l + b x'_j(1 - \gamma)}{(x'_j)^2(1 - \gamma)^2},$$

where

$$\Delta l \triangleq l_j(\alpha_{\text{min}}; \bar{y}) - l_j(0; \bar{y}),$$

where $l_j(\alpha; \bar{y})$ is computed from equation (17.35). Notice the expression for $l_j(\alpha; \bar{y})$ contains logarithms so its evaluation can be more computationally expensive.

An alternative approach is to use the simplified surrogate function of Theorem 17.3.2 based on the expression for $c$ in equation (17.30). In this case, $c$ is given by

$$c = \frac{l'_j(0; \bar{y}) - l'_j(\alpha_{\text{min}}; \bar{y})}{x'_j(1 - \gamma)},$$

where $l'_j(\alpha; \bar{y})$ is computed from equation (17.36). Again, putting this all together results in the general ICD update algorithm given in Figure 17.7.

### 17.4 Poisson Surrogate Based on EM Algorithm

While the Poisson negative log likelihood function is convex, minimization can still be quite tricky since this is not a quadratic function of $x$. The previous section presented some techniques for directly constructing surrogate functions to the Poisson log likelihood.

However, another approach to forming a surrogate function is to use a method based on the EM algorithm that was first pioneered by Shepp and Verdi [60]. Shepp and Verdi’s method works by using the full set of unknown counts, $Z_{i,j}$, as the complete information, and then using the subset of information, $Y_i$, as the incomplete information.

Referring back to Section 17.1 we can think of each $Z_{i,j}$ as being photons independently generated at a conditional rate of $A_{i,j}X_j$. Then the actual
ICD for Poisson Measurements $Y|X \sim \text{Pois}(AX)$:

- Initialize $0 < \gamma < 1$
- Initialize $x > 0$
- $\bar{y} \leftarrow Ax$
- For $K$ iterations {
  - For each pixel $j \in 1,\cdots,N$ {
    - $x_{\text{min}} \leftarrow -(1 - \gamma)x_j$
    - $b \leftarrow l'_j(0; \bar{y})$ using equation (17.36)
    - $c \leftarrow \frac{l'_j(0; \bar{y}) - l'_j(x_{\text{min}}; \bar{y})}{x_j(1 - \gamma)}$ using equation (17.36)
    - $\hat{\alpha} \leftarrow \arg \min_{\alpha \geq \alpha_{\text{min}}} \left\{ b\alpha + \frac{c}{2}\alpha^2 + u_j(\alpha + x_j, x_{\bar{j}}) \right\}$
    - $\bar{y} \leftarrow \bar{y} + A_{*j}\hat{\alpha}$
    - $x_j \leftarrow x_j + \hat{\alpha}$
  }
}

Figure 17.7: Pseudocode for ICD computation of the MAP estimate with a Poisson forward model of the form $Y|X \sim \text{Pois}(AX)$ and a general prior model. Each ICD update assumes the use of a quadratic surrogate function for the Poisson log likelihood. The parameter $0 < \gamma < 1$ determines the minimum allowed value of the pixel update.

observations, $Y_i$, are formed by

$$Y_i = \sum_{j=1}^{N} Z_{i,j};$$

so that it then has a conditional distribution given by $Y_i|X \sim \sum_{i=1}^{N} A_{i,j}X_j$. This also implies that the complete set of photon counts has conditional distribution $Z_{i,j}|X \sim \text{Pois}(A_{i,j}X_j)$.

Given this framework, our objective is to derive a surrogate function for the negative log likelihood function $l(x) = -\log p(y|x)$. To do this, we will adapt the $Q$ function of the EM algorithm from Chapter 12 equation (12.11). Recall that the function $Q(x; x')$ of the EM algorithm is a surrogate function for the maximization of the log likelihood $\log p(y|x)$. So given this fact, our
17.4 Poisson Surrogate Based on EM Algorithm

Approach will be to calculate an expression for the $Q$ function, and then define a general surrogate function for the negative log likelihood as

$$l(x; x') = -Q(x; x') + \text{const} , \quad (17.41)$$

where const is anything that is not a function of $x$. We can do this by using the following formula for the $Q$ function adapted from Chapter 12 equation (12.11).

$$Q(x; x') = \mathbb{E}[\log p(Z|x)|Y = y, x'] \quad (17.42)$$

Using the log likelihood expression of equation (17.15), we can substitute into the expression of equation (17.42).

$$Q(x; x') = \mathbb{E}[\log p(Z|x)|Y = y, x']$$

where

$$\bar{Z}_{i,j} = \mathbb{E}[Z_{i,j}|Y = y, x']$$

Now in order to simplify the expression for $\bar{Z}_{i,j}$, we will first need to define some additional terms. Define

$$Z_{i,j}^{(c)} = \sum_{k \neq j} Z_{i,k} = Y_i - Z_{i,j} \quad , \quad (17.43)$$

so that $Y_i = Z_{i,j} + Z_{i,j}^{(c)}$ where $Z_{i,j}$ and $Z_{i,j}^{(c)}$ are independent with conditional means given by

$$\lambda_1 \triangleq \mathbb{E}[Z_{i,j}|X] = A_{i,j}X_j \quad (17.44)$$

$$\lambda_2 \triangleq \mathbb{E}[Z_{i,j}^{(c)}|X] = \sum_{k \neq j} A_{i,k}X_k \quad . \quad (17.45)$$
So using the facts that $Y_i = Z_{i,j} + Z_{i,j}^{(c)}$ and that $Z_{i,j}$ is independent of $Y_k$ for $k \neq i$, then we have that

$$\bar{Z}_{i,j} = \mathbb{E}[Z_{i,j} | Y = y, x']$$

$$= \mathbb{E}[Z_{i,j} | Y_i = y_i, x']$$

$$= \mathbb{E}[Z_{i,j} | Z_{i,j} + Z_{i,j}^{(c)} = y_i, x']$$

Now in order to evaluate this last expression for $\bar{Z}_{i,j}$, we first derive an explicit expression for the joint conditional PMF of $\bar{Z}_{i,j}$ and $Z_{i,j}^{(c)}$ given $X$.

$$p_{z,z^{(c)}|x}(k, l|x') \triangleq P\{ Z_{i,j} = k, Z_{i,j}^{(c)} = l | X = x' \}$$

$$= \frac{\lambda_1^k e^{-\lambda_1} \lambda_2^l e^{-\lambda_2}}{k! l!}$$

From this we know that

$$p_{z,z^{(c)}|y,x}(k, l|y_i, x') \triangleq P\{ Z_{i,j} = k, Z_{i,j}^{(c)} = l | Z_{i,j} + Z_{i,j}^{(c)} = y_i, X = x' \}$$

$$= 1/z p_{z,z^{(c)}|x}(k, l|x') \delta(k + l = y_i)$$

where $z$ is the normalizing constant required so that $p_{z,z^{(c)}|y}(k, l|y_i)$ sums to 1. So from this we can see that

$$p_{z|y,x}(k|y_i, x') \triangleq P\{ Z_{i,j} = k | Z_{i,j} + Z_{i,j}^{(c)} = y_i, X = x' \}$$

$$= P\{ Z_{i,j} = k, Z_{i,j}^{(c)} = y_i - k | Z_{i,j} + Z_{i,j}^{(c)} = y_i, X = x' \}$$

$$= 1/z p_{z,z^{(c)}|x}(k, y_i - k|x').$$

From this point, it is simply a matter of algebra to show the result that

$$p_{z|y,x}(k|y_i, x') = 1/z p_{z,z^{(c)}}(k, y_i - k|x')$$

$$= \left( \frac{y_i}{k} \right) \left( \frac{\lambda_1}{\lambda_1 + \lambda_2} \right)^k \left( \frac{\lambda_2}{\lambda_1 + \lambda_2} \right)^{y_i-k}.$$  \hspace{0.5cm} (17.46)

The form of equation (17.46) is the binomial distribution with parameters $y_i$ and

$$p = \frac{\lambda_1}{\lambda_1 + \lambda_2}.$$
So we denote this by \( Z_{i,j} | (Y, X) \sim B(y_i, p) \). From this we can calculate the conditional expectation required for \( \bar{Z}_{i,j} \) as

\[
\bar{Z}_{i,j} = y \frac{\lambda_1}{\lambda_1 + \lambda_2} = y \frac{A_{i,j} x_j'}{\sum_{j=1}^N A_{i,j} x_j'}
\]

Now going back to equation (17.41), we have that

\[
l(x; x') = -Q(x; x') + \text{const} = \sum_{i=1}^M \sum_{j=1}^N \{ A_{i,j} x_j - \bar{Z}_{i,j} \log(A_{i,j} x_j) \} + \text{const}
\]

\[
= \sum_{j=1}^N \sum_{i=1}^M \{ A_{i,j} x_j - \bar{Z}_{i,j} \log(x_j) \} + \text{const}
\]

\[
= \sum_{j=1}^N \left\{ \left( \sum_{i=1}^M A_{i,j} \right) x_j - \left( \sum_{i=1}^M \bar{Z}_{i,j} \right) \log(x_j) \right\} + \text{const}
\]

\[
= \sum_{j=1}^N \{ \alpha_j x_j - \beta_j \log(x_j) \}
\]

where

\[
\alpha_j = \sum_{i=1}^M A_{i,j}
\]

\[
\beta_j = x_j' \sum_{i=1}^M \frac{y_i A_{i,j}}{\sum_{k=1}^N A_{i,k} x_k'}
\]

So in summary, a surrogate function for the negative log likelihood of the Poisson distribution given by

\[
l(x) = \sum_{i=1}^M \{ A_{i,*} x - y_i \log(A_{i,*} x) + \log(y_i!) \}
\]

(17.47)

is given by

\[
l(x; x') = \sum_{j=1}^N \{ \alpha_j x_j - \beta_j \log(x_j) \}
\]
where

$$
\alpha_j = \sum_{i=1}^{M} A_{i,j}
$$

$$
\beta_j = x'_j \sum_{i=1}^{M} \frac{y_i A_{i,j}}{\sum_{k=1}^{N} A_{i,k} x'_{k}}
$$

The crucial value of this result is that the surrogate function, $l(x; x')$ is a sum of decoupled terms index by $j$ the pixel index. This means that the EM surrogate function appears as if the Poisson measurements are made independently at each pixel!

Putting this all together, Figure 17.8 shows the final algorithm resulting from the integration of the EM algorithm surrogate function with ICD optimization for the M-step using the algorithm of Figure 17.6.
EM Optimization for Poisson Measurements $Y|X \sim \text{Pois}(AX)$:

Initialize $0 < \gamma < 1$
Initialize $x > 0$
For $K$ iterations {
  For each pixel $j \in 1, \cdots, N$ {
    $\alpha_j \leftarrow \sum_{i=1}^{M} A_{i,j}$
    $\beta_j \leftarrow x_j \sum_{i=1}^{M} \frac{y_i A_{i,j}}{\sum_{k=1}^{N} A_{i,k} x_k}$
  }
  For each pixel $j \in 1, \cdots, N$
  $x_{\text{min}} \leftarrow \gamma x_j$
  $b \leftarrow \alpha_j - \beta_j / x_j$
  $c \leftarrow \frac{\beta_j}{\gamma x_j^2}$
  $x_i \leftarrow \arg \min_{\alpha \geq x_{\text{min}}} \left\{ b(\alpha - x_j) + \frac{c}{2}(\alpha - x_j)^2 + u_j(\alpha, x_{\partial j}) \right\}$
}

Figure 17.8: Pseudocode for EM computation of the MAP estimate with a Poisson forward model of the form $Y|X \sim \text{Pois}(AX)$ and a general prior model. This algorithm uses the EM surrogate function together with the ICD optimization algorithm of Figure [17.6].

17.5 Chapter Problems

1. Calculate the sum, mean, and variance of the PMF for the Poisson distribution given in equations (17.2), (17.3), and (17.4).

2. Calculate the ML estimate of $X$ given $Y$ of equation (17.6).

3. Let $Y = \sum_{i=1}^{N} Z_j$ be the sum of $N$ independent Poisson random variables each with $Z_j \sim \text{Pois}(\lambda_j)$. Show that $Y$ is then Poisson with mean $\lambda = \sum_{i=1}^{N} \lambda_j$.

4. Derive the ML estimate of $\lambda$ given $Y$ of equation (17.6).
5. Show that the negative log likelihood function of equation (17.17) is convex.

6. Let $Y|X \sim \text{Pois}(X)$ where $X$ and $Y$ are both scalars, so that

$$ l(y|x) = - \log p(y|x) = x - y \log x + \log(y!) . $$

a) Show that for all $y \in \mathbb{Z}^+$ that $l(y|x)$ is a convex function of $x$.
b) Show that for all $y \in \mathbb{Z}^+$ that $l'(y|x)$ is a concave function of $x$ where $l'(y|x) \triangleq \frac{\partial l(y|x)}{\partial x}$.

7. Consider the surrogate function $Q(x; x')$ defined in Theorem 17.3.1. Furthermore, define

$$ q(x; x') = Q(x; x') + f(x') . $$

Show that the following are true.
a) Show that $q(x'; x') = f(x')$.
b) Show that $\left. \frac{\partial q(x; x')}{\partial x} \right|_{x=x'} = b = f'(x')$.
c) Show that $q(x_{\text{min}}; x') = f(x_{\text{min}})$.

8. Prove that $Z_{i,j}$ and $Z_{i,j}^{(c)}$ defined in equation (17.43) are independent with conditional means given by equations (17.44) and (17.45).
Appendix A

A Review of Convexity in Optimization

In order to better understand the use of convex potential functions in prior modeling, we first need to review some basic definitions and properties of convex functions and sets. We start with the definition of a convex set in $\mathbb{R}^N$.

**Definition:** A set $\mathcal{A} \subset \mathbb{R}^N$ is said to be a **convex set** if for all $x, y \in \mathcal{A}$ and for all $0 < \lambda < 1$,

$$\lambda x + (1 - \lambda)y \in \mathcal{A}.$$ 

So a convex set has the property that any line connecting two points in $\mathcal{A}$ is contained within the set $\mathcal{A}$. Convex sets occur commonly, and include sets such as $\mathbb{R}^N$ and $\mathbb{R}^+ = \{x \in \mathbb{R}^N : \forall i \ x_i \geq 0\}$. This latter set is very important because it is often used as a constraint on the solution of MAP estimates when $x$ is an image. Below are some important and useful properties of convex sets.

**Property A.1. Basic properties of convex sets** - Let $\mathcal{A}$ and $\mathcal{B}$ be convex subsets of $\mathbb{R}^N$; let $\mathcal{C}$ be a convex subset of $\mathbb{R}^M$; and let $A$ be a $M \times N$ matrix. Then the following properties hold.

a) The set $\mathcal{A} \cap \mathcal{B}$ is a convex set.

b) The set $\{x \in \mathbb{R}^N : Ax \in \mathcal{C}\}$ is convex.

c) The set $\mathbb{R}^N$ is a convex set.
Figure A.1: Figure a) illustrates an example of a non-convex function, \( f(x) = (x - 1)^2(x+1)^2 + x + 1.1 \), which contains a local minimum that is not global. Alternatively, Figure b) shows an example of a convex function, \( f(x) = e^{-x} \), which has no minima, either local or global. Finally, Figure c) shows the function \( f(x) = x^2 \), which is strictly convex with a local minimum, which must also be the global minimum.

d) The set \( \mathbb{R}^+ \) is a convex set.

One very important property of convex sets is stated in the following theorem.

**Theorem A.0.1. Hilbert Projection Theorem**

Let \( \mathcal{A} \subseteq \mathbb{R}^N \) be a convex set. Then for all \( x \in \mathbb{R}^N \) there exists a unique point \( y \in \mathcal{A} \) for which \( \|x - y\| \) is minimized over all points in \( \mathcal{A} \).

Using this result, we define the following notation for the projection of a vector \( x \) onto a non-empty convex set \( \mathcal{A} \)

\[
\hat{x} = \text{clip}\{x, \mathcal{A}\} = \arg \min_{y \in \mathcal{A}} \|x - y\|.
\]

By Theorem [A.0.1], we then know that this projection both exists and is unique. This notation will be useful when dealing with convex constraints such as positivity. For example, we can enforce positivity using the following notation

\[
\hat{x} = \text{clip}\{x, \mathbb{R}^+ \}
\]

where \( \hat{x} \) is then constrained to be in the non-negative quadrant of \( \mathbb{R}^N \).

Now that we have defined convex sets, we can use this definition to define convex functions.

**Definition:** Let \( \mathcal{A} \) be a convex subset of \( \mathbb{R}^N \). Then we say that a function \( f : \mathcal{A} \to \mathbb{R} \) is **convex** if for all \( x, y \in \mathcal{A} \) such that \( x \neq y \), and for all \( 0 < \lambda < 1 \),

\[
f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).
\]
Furthermore, we say that \( f \) is **strictly convex** if the inequality is strict.

Concave functions have a closely related definition.

**Definition:** Let \( \mathcal{A} \) be a convex subset of \( \mathbb{R}^N \). Then we say that a function \( f : \mathcal{A} \to \mathbb{R} \) is **(strictly) concave** on \( \mathcal{A} \) if the function \( -f \) is (strictly) convex on \( \mathcal{A} \).

Convexity is a very valuable property for two reasons. First, it is often easy to show that functions are concave/convex. This is primarily because the sum of convex functions is convex. Second, convex functions have important advantages in optimization problems. In particular, it is much easier to guarantee both existence of global minimum and convergence of iterative optimization algorithms when the function being minimized is convex. The following lists out some basic properties of convex functions that will be very useful. The corresponding results are, of course, also true for concave functions by applying these results to the function \( -f \).

**Property A.2. Basic properties of convex functions** - Let \( \mathcal{A} \) be a convex subset of \( \mathbb{R}^N \); let \( f : \mathcal{A} \to \mathbb{R} \) and \( g : \mathcal{A} \to \mathbb{R} \) be convex functions; let \( A \) be an \( N \times M \) matrix; and let \( B \) be an \( N \times N \) symmetric and positive semi-definite matrix. Then the following are true.

a) \( f(x) \) is a continuous function on \( \text{int}(\mathcal{A}) \).

b) \( h(x) = f(Ax) \) for \( x \in \{ x \in \mathbb{R}^M : Ax \in \mathcal{A} \} \) is a convex function.

c) \( h(x) = f(x) + g(x) \) for \( x \in \mathcal{A} \) is a convex function.

d) \( h(x) = \|x\|^2_B = x^tBx \) for \( x \in \mathcal{A} \) is a convex function.

All these properties are of great value of great value, but property a) above deserves some additional explanation in order to fully understand its implications. First, we know that if \( f \) is a convex function on \( \mathcal{A} = \mathbb{R}^N \), then \( f \) must be continuous since \( \text{int}(\mathbb{R}^N) = \mathbb{R}^N \). While this is a common case for us, it is important to remember that convex functions need not always be continuous. To see this, consider the counter example given by the function \( f : [0, 1] \to \mathbb{R} \) defined below.

\[
f(x) = \begin{cases} 1 & \text{if } x = 0 \\ x^2 & \text{if } x \in (0, 1] \end{cases}
\]
Notice that $f$ is a convex function defined on the convex set $[0, 1] \subset \mathbb{R}$; however, $f$ is not continuous at $x = 0$ since $\lim_{x \to 0} f(x) = 0 \neq 1$. Furthermore, $f$ does not take on a local or global minimum because it never actually achieves its infimum. Moreover, this convex function also has the property that for $\alpha = 1/2$ its sublevel set is not closed.

$$A_{1/2} = \{ x \in [0, 1] : f(x) \leq 1/2 \} = (0, \sqrt{1/2}]$$

This is an important counter example that we should keep in mind. Later we will introduce the concept of closed convex functions to guard against this sort of situation.

Of course, strict convexity is stronger than convexity, and in some cases, it is very useful. However, the conditions required for strict convexity are a bit different and are given below.

**Property A.3. Basic properties of strictly convex functions** - Let $\mathcal{A}$ be a convex subset of $\mathbb{R}^N$; let $f : \mathcal{A} \to \mathbb{R}$ be a strictly convex function; let $g : \mathcal{A} \to \mathbb{R}$ be a convex function; let $A$ be an $N \times M$ matrix of rank $M$; and let $B$ be an $N \times N$ symmetric and positive-definite matrix. Then the following are true.

a) $h(x) = f(Ax)$ for $x \in \{ x \in \mathbb{R}^M : Ax \in \mathcal{A} \}$ is a strictly convex function.

b) $h(x) = f(x) + g(x)$ for $x \in \mathcal{A}$ is a strictly convex function.

c) $h(x) = \|x\|_B^2 = x^t B x$ for $x \in \mathcal{A}$ is a strictly convex function.

In some cases, it is useful to show that a function is convex by showing it is the point-wise maximum of a set of functions.

**Property A.4. Point-wise maximum over sets of convex functions** - Let $f_1(x)$ and $f_2(x)$ be two convex functions, and let $g(x, y)$ be a convex function of $x$ for each value of $y \in \mathcal{B}$ where $\mathcal{B}$ is not necessarily a convex set. Then the following are true.

a) $h(x) = \max\{ f_1(x), f_2(x) \}$ is a convex function.

b) $h(x) = \sup_{y \in \mathcal{B}} \{ g(x, y) \}$ is a convex function.

When optimizing functions in high dimensional spaces, it is easy to become trapped in local minima. For example, Figure A.1a illustrates a case in
which a function has a local minimum that is not the global minimum. Local
minimum can make optimization very difficult by trapping optimization
algorithms in poor solutions.

Convex functions and sets are extremely useful concepts in optimization
because they can be used to ensure that any local minimum must be global,
thereby eliminating this potentially problematic case. To see this, consider
the strictly-convex function shown in Figure A.1b.

In order to make this more precise, we first must give precise definitions
to the concept of global and local minimum of a function. We can do this by
using the concept of an “epsilon ball” about a point $x^*.$

$$B(x^*, \epsilon) \triangleq \{ x \in \mathbb{R}^N : ||x - x^*|| < \epsilon \}$$

So, $x^*$ is a local minimum of a function $f(x)$ if there exists a sufficiently small
value of $\epsilon$ so that $f(x) \geq f(x^*)$ for all the points in the set $x \in B(x^*, \epsilon) \cap A.$

Definition: Consider a function $f : A \to \mathbb{R}$ where $A \subset \mathbb{R}.$ Then we
say that the function $f$ has a **local minimum** at $x^* \in A$ if $\exists \epsilon > 0,$ such that $\forall x \in B(x^*, \epsilon) \cap A$ it is the case that $f(x) \geq f(x^*).$

Alternatively, $x^*$ is a global minimum if it achieves the minimum value of
the function over the entire domain. So we have the following definition.

Definition: Consider a function $f : A \to \mathbb{R}$ where $A \subset \mathbb{R}^N.$ Then we say
that the function $f$ has a **global minimum** at $x^* \in A$ if $\forall x \in A$ it is
the case that $f(x) \geq f(x^*).$

Using these definitions, we can now state some very powerful theorems about
the nature of local and global minimum for convex functions.

**Theorem A.0.2. Equivalence of Local and Global Minimum for a Convex
Function**

Let $f : A \to \mathbb{R}$ be a convex function on a convex set $A \subset \mathbb{R}^N.$ Then $f$ has a
local minimum at $x^* \in A$ if and only if $f$ has a global minimum at $x^* \in A.$

This is a powerful theorem because it states that for convex functions, we
need only find local minimum of the function in order to be assured that
the minima are actually global. In fact, many of the iterative optimization
algorithms that we have seen operate locally on the function being minimized,
so without this theorem, these methods have little chance of obtaining a
global minimum for very complex functions.

However, while Theorem A.0.2 states that local and global minimum are
equivalent, it does not assure that they exist or that they are unique if they
do exist. The following theorem handles the issue of uniqueness when the
function being minimized is strictly convex.

**Theorem A.0.3. Uniqueness of Global Minimum for a Strictly Convex Function**

Let \( f: \mathcal{A} \to \mathbb{R} \) be a strictly convex function on a convex set \( \mathcal{A} \subset \mathbb{R}^N \). Then if \( f \) has a global minimum at \( x^* \in \mathcal{A} \) then that global minimum must unique
so that for all \( x \in \mathcal{A} \) such that \( x \neq x^* \), we have that \( f(x) > f(x^*) \).

Existence of a global minimum is a more difficult matter. In fact, it is easy
to come up with examples of strictly convex functions which have no global
minimum. For example, the function \( f(x) = e^{-x} \) shown in Figure A.1(b) is
such an example. In this case, there is no global minimum because the func-
tion decreases monotonically toward infinity, never achieving its minimum
value.

In order to assure the existence of a global minimum, we will need some
additional concepts defined below.

**Definition:** Let \( \mathcal{A} \subset \mathbb{R}^N \), then we say that \( \mathcal{A} \) is:

- **Closed** if every convergent sequence in \( \mathcal{A} \) has its limit in \( \mathcal{A} \).
- **Bounded** if \( \exists M \) such that \( \forall x \in \mathcal{A}, \|x\| < M \).
- **Compact** if \( \mathcal{A} \) is both closed and bounded.

With these definitions, we can invoke a powerful theorem from analysis which
assures us that any continuous function takes on its minimum and maximum
on a compact set [63].

**Theorem A.0.4. Weierstrass Extreme Value Theorem**

Let \( f: \mathcal{A} \to \mathbb{R} \) be a continuous function on a compact set \( \mathcal{A} \subset \mathbb{R}^N \). Then \( f \) takes on its global minimum in \( \mathcal{A} \), i.e., there exists an \( x^* \in \mathcal{A} \) such that
\( \forall x \in \mathcal{A}, f(x^*) \leq f(x) \).

Putting together the Extreme Value Theorem with the results of Theo-
rems A.0.2 and A.0.3 yields the following important result.
**Theorem A.0.5. Global Minimum of Convex Functions**

Let $f : \mathcal{A} \to \mathbb{R}$ be a strictly convex continuous function on a compact and convex set $\mathcal{A} \subset \mathbb{R}^N$. Then $f$ takes on a unique global minimum at some point $x^* \in \mathcal{A}$.

Notice that we must explicitly assume continuity of $f$ in Theorem A.0.5 because convex functions need not be continuous on the boundary of $\mathcal{A}$.

In many cases, we will be faced with the case where $\mathcal{A}$ is closed but not bounded, and therefore not compact. For example, if $\mathcal{A} = \mathbb{R}^N$ then $\mathcal{A}$ is both open and closed, but it is neither bounded nor compact. Nonetheless, even in this case, it can often be shown from the structure of the problem that a global minimum exists.

In order to better handle these cases, we define two properties of functions that will serve as handy tools. First, we define the sublevel set of a function.

*Definition:* The $\alpha$-sublevel set of a function is the set defined by

$$\mathcal{A}_\alpha = \{ x \in \mathbb{R}^N : f(x) \leq \alpha \} .$$

Next we define the concept of a compact sublevel set.

*Definition:* We say that a function $f(x)$ has a compact sublevel set if there exists an $\alpha \in \mathbb{R}$ such that the sublevel set $\mathcal{A}_\alpha$ is compact and non-empty.

Using this definition, it is easily shown that any strictly convex function with a compact sublevel set has a unique global minimum. Armed with these new definitions, we may state the following two corollaries of Theorems A.0.4 and A.0.5.

**Corollary 1. Minimum of Functions with Compact Sublevel Sets**

Let $f : \mathcal{A} \to \mathbb{R}$ be a continuous function on $\mathcal{A} \subset \mathbb{R}^N$ with a compact sublevel set. Then $f$ takes on a global minimum for some $x^* \in \mathcal{A}$, so that $\forall x \in \mathcal{A}, f(x^*) \leq f(x)$.

**Corollary 2. Global Minimum of Functions with Compact Sublevel Sets**

Let $f : \mathcal{A} \to \mathbb{R}$ be a strictly convex continuous function on a convex set $\mathcal{A} \subset \mathbb{R}^N$ with a compact sublevel set. Then $f$ takes on a unique global minimum for some $x^* \in \mathcal{A}$, so that $\forall x \in \mathcal{A}, f(x^*) < f(x)$. 
These two corollaries remove the need for the domain of a convex function to be compact in order to ensure that the global minimum exists and is unique. So for example, functions such as \( f(x) = \frac{1}{2}x^tHx + b^tx \) where \( H \) is a positive definite matrix are strictly convex but do not have a compact domain. Yet, this function is has a compact sublevel set, so it does take on a unique global minimum. (See Problem 2 of Chapter 7)

Finally, we introduce two standard definitions that are commonly used in conjunction with convex functions that take values on the extended real line that includes \( \infty \). We will mostly restrict ourselves to functions that only take values in \( \mathbb{R} \); so we will mostly avoid using these definitions, but we present them here both for completeness and for the special situations when they are useful.

First we give the definition for an extended convex function which takes values on \( \mathbb{R} \cup \{\infty\} \).

**Definition:** We say that the extended function \( f : \mathbb{R}^N \rightarrow \mathbb{R} \cup \{\infty\} \) is **convex** if for all \( x, y \in A \) and for all \( 0 < \lambda < 1 \),

\[
f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y),
\]

where for any \( a \in \mathbb{R} \cup \{\infty\} \), we define that \( \infty + a = \infty \) and that \( \infty \leq \infty \).

Now it is interesting to note that if \( f : A \rightarrow \mathbb{R} \) is a convex function on the convex set \( A \), then we always have the option of defining a new extended convex function given by

\[
\tilde{f}(x) = \begin{cases} 
  f(x) & \text{if } x \in A \\
  \infty & \text{otherwise}
\end{cases}, \quad (A.1)
\]

where it is easily verified that \( \tilde{f} \) is convex. (See problem 1) The point here is that using extended convex functions can sometimes simplify our lives since the domain of the function can always be \( \mathbb{R}^N \).

However, the downside of extended convex functions is that they can lead to some unintended technical difficulties. A proper convex function is closed if and only if it is lower semi-continuous. So in order to guard against these problems, mathematicians typically also assume that extended convex functions are also **proper and closed**. The following two definitions give meaning to these two terms.
**Definition:** We say the function \( f : \mathbb{R}^N \to \mathbb{R} \cup \{\infty\} \) is **proper** if there exists an \( x \in \mathbb{R}^N \) so that \( f(x) < \infty \).

So if a function is proper, then we know that it can be not infinite everywhere, which is typically a very reasonable assumption.

**Definition:** We say the function \( f : \mathbb{R}^N \to \mathbb{R} \cup \{\infty\} \) is **closed** if for all \( \alpha \in \mathbb{R} \), the sublevel set \( A_\alpha = \{ x \in \mathbb{R}^N : f(x) \leq \alpha \} \) is a closed set.

So if a function is proper, then we know that it can be not infinite everywhere, which is typically a very reasonable assumption.

Closure eliminates the possibility that very strange things can happen when minimizing a convex function. In particular, while it does not guarantee that a minimum exists, it does removed certain pathological cases. For example, consider the function

\[
f(x) = \begin{cases} 
\mathbf{1}^t x & \text{if } ||x|| < 1 \\
\infty & \text{if } ||x|| \geq 1 
\end{cases}
\]

where \( \mathbf{1} \) is a column vector of 1's. Then in this case \( f \) is not closed since \( \{ x \in \mathbb{R}^N : f(x) < \infty \} \) is an open set. However, even though the function is convex and tends to infinity as \( ||x|| \to \infty \), it does not take on a global or local minimum. This is because the minimum “should” occur at \( x = -\mathbf{1} \), but it doesn’t because \( f(-\mathbf{1}) = \infty \). Intuitively, the problem is that \( f \) is not continuous at \( x = -\mathbf{1} \).

We conclude with a result that provides some additional intuition about the nature of proper closed convex functions [4].

**Corollary 3. Continuity of Proper Close Convex Functions**

Let \( f : \mathbb{R}^N \to \mathbb{R} \cup \{\infty\} \) be a proper convex function. Then \( f \) is closed if and only if it is lower semi-continuous.

So for a proper convex function, closure is equivalent to lower semi-continuity, which is very valuable when minimizing functions.
A.1 Appendix Problems

1. Prove statements of Property A.1.

2. Give a counter-example to the following statement. “If $\mathcal{A}$ and $\mathcal{B}$ are convex sets, then $\mathcal{A} \cup \mathcal{B}$ is a convex set.”

3. Prove the statements a), b), and c) of Property A.3.

4. Prove the statements d) of Property A.3; that is, prove if $f: \mathcal{A} \to \mathbb{R}$ is a convex function on the convex set $\mathcal{A} \subset \mathbb{R}^N$, then it is a continuous function for all $x \in \mathcal{A}$.

5. Prove the statements of Property A.3.

6. Prove the Property A.4a.

7. Prove Theorem A.0.2.

8. Prove Theorem A.0.3.

9. Prove Theorem A.0.5.
Appendix B

The Boltzmann and Gibbs Distributions

In this appendix, we derive the general form of the discrete distribution known as the Boltzmann or Gibbs distribution, and also derive some of its important properties.

Let \( X \) be a discrete valued random variable with PMF given by \( p_x = P\{X = x\} \). In practice, \( X \) can be an array of discrete random variables, but the key issue is that \( X \) takes on values in a discrete set. With each value or state, \( x \), we will associate an energy denoted by \( u(x) \). So then the expected energy is given by

\[
\mathcal{E} = \mathbb{E}[u(X)] = \sum_x u(x) p_x ,
\]

and the entropy of the random variable \( X \) is then defined as

\[
\mathcal{H} = \mathbb{E}[-\log p_X] = \sum_x -p_x \log p_x .
\]

The first law of thermodynamics states that energy is conserved for a closed system. So the expected energy, \( \mathcal{E} \), must be constant. In addition, the second law of thermodynamics states that the entropy of a closed system can only increase with time. Therefore, a closed system in thermodynamic equilibrium is one that has achieved its maximum entropy; so that no further increases in entropy can occur and the system is in a stable equilibrium.

Therefore, the Boltzmann distribution can be derived by maximizing the entropy, \( \mathcal{H} \), subject to the constraint that the energy is constant, i.e., \( \mathcal{E} = \mathcal{E}_o \). Based on this logic, we can derive the Boltzmann distribution by computing
the values of the discrete probabilities, \( p_x \), that maximize \( \mathcal{H} \) subject to the two constraints that \( \mathcal{E} \) is constant and \( \sum_x p_x = 1 \).

Using the method of Lagrange multipliers, we can change the constrained optimization problem into an unconstrained optimization problem. Then the solution to the optimization problem can be found by solving the equation

\[
\frac{\partial}{\partial p_x} \left[ \mathcal{H} - \beta (\mathcal{E} - \mathcal{E}_o) + \lambda \left( \sum_x p_x - 1 \right) \right] = 0 \quad (\text{B.1})
\]

where \(-\beta\) and \(\lambda\) are the Lagrange multiplier constants for the two constraints. In order to evaluation this expression, we first compute derivatives of the energy and entropy expressions.

\[
\frac{\partial}{\partial p_x} \mathcal{E} = \frac{\partial}{\partial p_x} \sum_k p_k u(k) = u(x)
\]

\[
\frac{\partial}{\partial p_x} \mathcal{H} = \frac{\partial}{\partial p_x} \sum_k -p_k \log p_k = -1 - \log p_x
\]

Notice that the derivative is only non-zero for the term in which \( k = x \). Using these expressions, then equation (B.1) is given by

\[
-1 - \log p_x - \beta u(x) + \lambda = 0 . \quad (\text{B.2})
\]

Then solving for the value \( p_x \) and applying the constraint that \( \sum_x p_x = 1 \) results in an explicit form for the density given by

\[
p_x = \frac{1}{z_\beta} \exp \{-\beta u(x)\} , \quad (\text{B.3})
\]

where \( z_\beta \) is the partition function for the distribution given by

\[
z_\beta = \sum_x \exp \{-\beta u(x)\} .
\]

In statistical thermodynamics, the Lagrange multiplier is defined as \( \beta = \frac{1}{k_b T} \), where \( k_b \) is Boltzmann’s constant and \( T \) is the temperature in degrees Kelvin. Using this convention, we may rewrite the expression of (B.3) in the standard form

\[
p_x = \frac{1}{z_\beta} \exp \{-\beta u(x)\} , \quad (\text{B.4})
\]
where $\beta = \frac{1}{k_b T}$.

With this expression, we can now view the probability density as a function of $\beta$, a parameter inversely proportional to temperature. So notationally, we may express this as

$$p_\beta(x) \triangleq p_x,$$

where the notation $p_\beta(x)$ emphasizes the dependence on the inverse temperature $\beta$.

Consequently, the energy and entropy of the system can also be viewed as functions of $\beta$, resulting in the notation $\mathcal{E}_\beta$ and $\mathcal{H}_\beta$. With this in mind, we can derive the following three important properties of systems in thermodynamic equilibrium derived below in Section B.1.

$$\mathcal{H}_\beta = \beta \mathcal{E}_\beta + \log z_\beta$$  \hspace{1cm} (B.5)

$$\frac{d \log z_\beta}{d \beta} = -\mathcal{E}_\beta$$  \hspace{1cm} (B.6)

$$\frac{d \mathcal{H}_\beta}{d \beta} = \beta \frac{d \mathcal{E}_\beta}{d \beta}.$$  \hspace{1cm} (B.7)

So using these results, we can see that the physical temperature is proportional to the ratio of the change in energy per unit change in entropy.

$$T = \frac{1}{k_b} \frac{d \mathcal{E}}{d \mathcal{H}}$$

Generally, the entropy of a system tends to increase as the energy increases, so generally the physical temperature tends to be positive. But that said, it is physically possible to have negative temperatures when the number of high energy states in a system is limited. In this case, increasing the energy in a system can actually reduce the number of possible available states, and therefore reduce the entropy of the system. When this happens, the temperature can become negative.
B.1 Derivation of Thermodynamic Relations

The following derives the relationship of equation (B.5).

\[ H_\beta = \sum_x -p_x \log p_x \]
\[ = \sum_x p_x \{ \beta u(x) + \log z_\beta \} \]
\[ = \beta E_\beta + \log z_\beta . \]

The following derives the relationship of equation (B.6).

\[ \frac{d \log z_\beta}{d \beta} = \frac{1}{z_\beta} \frac{dz_\beta}{d \beta} \]
\[ = \frac{1}{z_\beta} \frac{d}{d \beta} \sum_x \exp \{-\beta u(x)\} \]
\[ = -\sum_x u(x) \frac{1}{z_\beta} \exp \{-\beta u(x)\} \]
\[ = -E_\beta \]

The differentiating equation (B.5) results in the relationship of equation (B.7).

\[ \frac{dH_\beta}{d \beta} = E_\beta + \beta \frac{dE_\beta}{d \beta} + \frac{d \log z_\beta}{d \beta} \]
\[ = E_\beta + \beta \frac{dE_\beta}{d \beta} - E_\beta \]
\[ = \beta \frac{dE_\beta}{d \beta} \]
Appendix C

Proofs of Poisson Surrogates

The following is a proof of Theorem 17.3.1 from Chapter 17.

Proof. In order to simplify the notation of the proof, we may with out loss of generality assume that \( x' = 0 \) and \( f(x') = 0 \). In this case, it must then be that \( x_{\text{min}} < 0 \) and \( b = 0 \). Also, since \( f(x) \) is convex and \( f'(x) \) is concave both must also be continuous functions of \([x_{\text{min}}, \infty)\).

Now in this case, \( Q(0; 0) = f(0) = 0 \), so in order to show that \( Q \) is a surrogate for the minimization of \( f(x) \), we need only show that for all \( x \in [x_{\text{min}}, \infty) \) that \( Q(x; 0) \geq f(x) \). To do this, we start by defining the convex function \( g(t) = ct - f'(t) \). Then we can show that

\[
Q(x; x') - f(x) = \frac{c}{2}x^2 - \int_0^x f'(t)dt
= \int_0^x (ct - f'(t)) dt
= \int_0^x g(t)dt .
\]

By the specific choice of \( c \) from equation (17.21), it is easily shown that

\[
f(x_{\text{min}}) = Q(x_{\text{min}}, x') ;
\]

so we know that

\[
\int_0^{x_{\text{min}}} g(t)dt = Q(x_{\text{min}}, x') - f(x_{\text{min}}) = 0 .
\]
Furthermore define the two functions
\[ G_1(x) = \int_0^x g(t)dt \]
\[ G_2(x) = \int_x^{x_{\min}} g(t)dt . \]

Then we know that for \( x \in [x_{\min}, 0] \)
\[ G_1(x) + G_2(x) = \int_0^{x_{\min}} g(t)dt = 0 . \]

Now if we can show that \( G_1(x) \geq 0 \) for all \( x \in [x_{\min}, \infty) \), then we know that
\[ Q(x; x') - f(x) = G_1(x) \geq 0 , \] (C.1)
and we will have proved the result that \( Q \) is a surrogate for the minimization of \( f(x) \). Furthermore, since \( Q(x_{\min}; x') - f(x_{\min}) = 0 \), then any smaller value of the coefficient \( c' < c \) will violate the surrogate function relationship. So therefore, if \( Q \) is a surrogate function for the minimization of \( f(x) \) then it must be the tightest quadratic surrogate.

Next, we will show that \( g_0 \triangleq g(x_{\min}) \geq 0 \). Since \( g(x) \) is a convex function with \( g(x_{\min}) = g_0 \) and \( g(0) = 0 \), we know that \( g(t) \leq g_0 \frac{t}{x_{\min}} \) for all \( t \in [x_{\min}, 0] \). Therefore we have that
\[ 0 = \int_0^{x_{\min}} g(t)dt = \int_0^0 g(t)dt \leq \int_0^{x_{\min}} g_0 \frac{t}{x_{\min}}dt = g_0 \frac{1}{2} . \]

So therefore, we know that \( g_0 \geq 0 \). Now in order to prove the theorem, we will consider two cases. Case 1 when \( g_0 = 0 \) and case 2 when \( g_0 > 0 \).

In case 1 when \( g_0 = 0 \), then since \( g(t) \) is convex with \( g(0) = 0 \), then it must be that \( g(t) \leq 0 \) for \( t \in [x_{\min}, 0) \). But since \( g(t) \) is continuous with \( \int_0^{x_{\min}} g(t)dt = 0 \), this implies that \( g(t) = 0 \) for \( t \in [x_{\min}, 0) \), which implies that for all \( x \in [x_{\min}, 0] \)
\[ G_1(x) = \int_0^x g(t)dt = \int_0^0 0dt = 0 . \]

Also, since \( g(t) \) is convex and \( g(x_{\min}) = g(0) = 0 \), it must be that \( g(x) \geq 0 \) for all \( x \in [0, \infty) \), which implies that for all \( x \in [0, \infty] \)
\[ G_1(x) = \int_0^x g(t)dt \geq 0 . \]
So for case 1 we have the result that $G_1(x) \geq 0$ for all $x \in [x_{\text{min}}, \infty)$.

In case 2, we assume $g_0 > 0$. Since the function $g(t)$ is continuous with $\int_{x_{\text{min}}}^0 g(t)dt = 0$, there must exist a point $x_1 \in (x_{\text{min}}, 0)$ such that $g(x_1) < 0$. Therefore, by the intermediate value theorem, there must exist a point $x_0 \in (x_{\text{min}}, x_1)$ such that $g(x_0) = 0$.

Since $g(t)$ is convex, then the following results must hold.

R1 $g(x) \geq 0$ for all $x \in [x_{\text{min}}, x_0]$

R2 $g(x) \leq 0$ for all $x \in [x_0, 0]$

R3 $g(x) \geq 0$ for all $x \in [0, \infty)$

Now result R3 implies that for all $x \in [0, \infty]$

$$G_1(x) = \int_0^x g(t)dt \geq 0.$$  

Result R2 implies that for all $x \in [x_0, 0]$

$$G_1(x) = \int_0^x g(t)dt = -\int_x^0 g(t)dt \geq 0.$$  

Result R1 implies that for all $x \in [x_{\text{min}}, x_0]$

$$G_1(x) = -G_2(x) = -\int_x^{x_{\text{min}}} g(t)dt = \int_{x_{\text{min}}}^x g(t)dt \geq 0.$$  

So for case 2 we have again shown the result that $G_1(x) \geq 0$ for all $x \in [x_{\text{min}}, \infty)$.

The following is a proof of Theorem 17.3.2 from Chapter 17.

Proof. In order to simplify the notation of the proof, we may with out loss of generality assume that $x' = 0$ and $f(x') = 0$. In this case, it must then be that $x_{\text{min}} < 0$ and $b = 0$. Also, since $f(x)$ is convex and $f'(x)$ is concave both must also be continuous functions of $[x_{\text{min}}, \infty)$.

Now in this case, $Q(0; 0) = f(0) = 0$, so in order to show that $Q$ is a surrogate for the minimization of $f(x)$, we need only show that for all
$x \in [x_{\text{min}}, \infty)$ that $Q(x; 0) \geq f(x)$. To do this, we start by defining the convex function $g(t) = ct - f'(t)$. Then we can show that

$$Q(x; x') - f(x) = \frac{c}{2}x^2 - \int_0^x f'(t)dt$$

$$= \int_0^x (ct - f'(t)) dt$$

$$= \int_0^x g(t)dt .$$

Furthermore if we define the function

$$G_1(x) = \int_0^x g(t)dt ,$$

then if we can show that $G_1(x) \geq 0$ for all $x \in [x_{\text{min}}, \infty)$, then we know that

$$Q(x; x') - f(x) = G_1(x) \geq 0 ,$$

and we have proved the result that $Q$ is a surrogate for the minimization of $f(x)$.

By the specific choice of $c$ from equation (17.21), it is easily show that $g(x_{\text{min}}) = 0$. Therefore, since $g(t)$ is convex and $g(0) = 0$, we know that the following results must hold.

R1  $g(x) \leq 0$ for all $x \in [x_{\text{min}}, 0]$

R2  $g(x) \geq 0$ for all $x \in [0, \infty)$

Now result R2 implies that for all $x \in [0, \infty]$

$$G_1(x) = \int_0^x g(t)dt \geq 0 .$$

Result R1 implies that for all $x \in [x_{\text{min}}, 0]$

$$G_1(x) = \int_0^x g(t)dt = -\int_x^0 g(t)dt \geq 0 .$$

So we have shown the result that $G_1(x) \geq 0$ for all $x \in [x_{\text{min}}, \infty)$.
Appendix D

Notation

General Notation:

1. Reals - \( \mathbb{R} \)
2. Integers - \( \mathbb{Z} \)
3. Non-negative Reals - \( \mathbb{R}^+ \)
4. \( N \) dimensional Non-negative Reals - \( \mathbb{R}^{+N} \)
5. Non-negative Integers - \( \mathbb{Z}^+ \)
6. Gradient is a column vector!
7. Vector norm \( ||x|| \)
8. Definition \( \triangleq \)
9. Expectation \( \mathbb{E}[X] \)
10. Expectation \( \mathbb{E}_\theta[X] \)
11. Variance \( \text{Var}[X] \)
12. Circular convolution \( x_n \odot h_n \)
13. Column vector of 1’s, \( \mathbf{1} \).
14. Column vector of 0’s, \( \mathbf{0} \).
15. Approximately equal \( x \approx y \)
16. Interior of set \( \text{int}(S) \)

17. \( \text{clip}\{x, [a, b]\} \) - clips the value \( x \) to the interval \([a, b]\)

18. \( \text{clip}\{x, \Omega\} \) - projects the vector \( x \) onto the convex set \( \Omega \).

19. \( \text{clip}\{x, \mathbb{R}^{+N}\} \) - enforce positivity constraint on \( x \).

20. \text{const} - additive constant
Probability (Chapter 2) Notation:

- $Y$ or $X \in \mathbb{R}^p$ - multivariate Gaussian observations
- $X_1, \ldots, X_n$ - i.i.d. samples from a distribution
- $P(A)$ - Probability of an event $A \subseteq \Omega$
- $P\{X \leq t\}$ - Probability of the event $A = \{\omega \in \Omega : X(\omega) \leq t\}$
- $P\{\frac{X}{1+Y} \leq t\}$ - Probability of the event $A = \left\{\omega \in \Omega : \frac{X(\omega)}{1+Y(\omega)} \leq t\right\}$
Gaussian prior models (Chapters 3 and 4) Notation:

- \( n \in [1, \cdots, N] \) - 1-D index set
- \( s = (s_1, s_2) \in [1, \cdots, N]^2 \) - 2-D index set with \( s_1 \) being the row and \( s_2 \) being the column.
- \( s \in S \) - arbitrary index set
- \( N = |S| \) - number of elements in lattice
- \( X_n \) - random process in 1 or more dimensions
- \( \mathcal{E}_s \) - causal or non-causal prediction error

- Causal notation
  - \( H \) - causal predictor matrix
  - \( A = I - H \) - causal predictor matrix
  - \( \mathcal{E} = AX = (I - H)X \) - causal prediction error
  - \( \Lambda \) - diagonal matrix of causal prediction variances
  - \( h_{i,j} \) or \( h_{i-j} \) - causal prediction filter
  - \( \mathcal{F}_n = \{X_i : i < n\} \) - past observations
  - \( H(\omega) \) - DTFT of causal prediction filter

- Non-causal notation
  - \( G \) - non-causal predictor matrix
  - \( B = I - G \) - non-causal predictor matrix
  - \( \mathcal{E} = BX = (I - G)X \) - non-causal prediction error
  - \( \Gamma \) - diagonal matrix of non-causal prediction variances
  - \( g_{i,j} \) or \( g_{i-j} \) - non-causal prediction filter
  - \( G(\omega) \) - DTFT of causal prediction filter
MAP Estimation with Gaussian priors (Chapters 5) Notation:

- $Y \in \mathbb{R}^N$ - Measured data
- $X \in \mathbb{R}^N$ - Unknown Image
- $A \in \mathbb{R}^{N \times N}$ - System matrix
- $W \sim N(0, \sigma^2 I)$ - Additive noise vector
- $\varepsilon_s$ - unit vector in direction of $s^{th}$ coordinate.
- $f(x)$ - MAP cost function
- $\nabla f(x)$ - gradient of MAP cost function
Non-Gaussian prior models (Chapters [6]) Notation:

- $\rho(\Delta)$ - potential function
- $\rho'(\Delta)$ - influence function
- $\rho''(\Delta)$ - derivative of influence function
MAP Estimation with non-Gaussian priors (Chapters 7) Notation:

- \( Y \in \mathbb{R}^M \) - Measured data
- \( X \in \mathbb{R}^N \) - Unknown Image
- \( X \in \mathbb{R}^{+N} \) - positivity constraint on unknown Image
- \( X \in \Omega \) - general convex constraint on unknown image
- \( A \in \mathbb{R}^{M \times N} \) - System matrix
- \( W \sim N(0, \sigma^2 I) \) - Additive noise vector
- \( Q(x; x') \) - surrogate function for \( f(x) \)
- \( \rho(\Delta; \Delta') \) - surrogate function for potential function \( \rho(\Delta) \)
Constrained Optimization for MAP Estimation (Chapters 9) Notation:

- $x \in \Omega \subset \mathbb{R}^N$ - Unknown Image where $\Omega$ is closed with non-empty interior
- $\lambda \in \mathcal{D} \subset \mathbb{R}^K$ - Lagrange parameter where $\mathcal{D}$ is convex
EM Algorithm (Chapter 12) Notation:

- $1 \leq n \leq N$ - time index and range
- $0 \leq m < M$ - state index and range
- $\mathbb{R}^p$ - observation space
- $Y_n \in \mathbb{R}^p$ for $n = 1, \cdots, N$ - observed data and associated space
- $X_n \in \Omega$ with $|\Omega| = M$ - unobserved data and associated space
- $\theta^{(k)}$ - result of $k^{th}$ EM iteration
- $N_m, b_m, R_m$ - sufficient statistics for each class
- $\bar{N}_m, \bar{b}_m, \bar{R}_m$ - expected sufficient statistics for each class
- $p$ - dimension of observation vector
- $\theta, \bar{\theta}$ - new and old parameters respectively
- $\pi_m, \mu_m, \sigma^2_m$ - parameters for $m^{th}$ mixture component
- $\theta = [\pi_0, \mu_0, \sigma^2_0, \cdots, \pi_{M-1}, \mu_{M-1}, \sigma^2_{M-1}]$ - full parameter vector
Markov Chain and HMM (Chapter 13) Notation:

- \(X_n\) - discrete time Markov chain
- \(0 \leq n \leq N\) with \(X_0\) being the initial state
- \(\pi_j^{(n)} = P\{X_n = j\}\) - state distribution at time \(n\)
- \(P_{i,j}^{(n)}\) - nonhomogeneous state transition probability
- \(\tau_j = \pi_j^{(0)}\) - initial state distribution
- \(N_j = \delta(X_0 - j)\) - initial state statistics
- \(K_j = \sum_{n=1}^{N} \delta(X_n - j) = \sum_{i=0}^{M-1} K_{i,j}\) - state membership statistics
- \(K_{i,j} = \sum_{n=1}^{N} \delta(X_n - j)\delta(X_{n-1} - i)\) - state transition statistics
- \(b_j = \sum_{n=0}^{N-1} y_n \delta(x_n - j)\) - first order class statistics
- \(S_j = \sum_{n=0}^{N-1} y_n y_n^t \delta(x_n - j)\) - second order class statistics
- \(\Omega = [0, \ldots, M - 1]\) - set of discrete states
- \(\theta = [\tau_j, P_{i,j} : \text{for } i, j \in \Omega]\) - parameter vector
- \(\pi_j\) - ergodic distribution of Markov chain
General MRF Models (Chapter 14) Notation:

- $\mathcal{T}$ - temperature in Gibbs distribution
- $\mathcal{C}$ - set of cliques in MRF
- $\mathcal{C}_s$ - set of cliques which contain the pixel $x_s$
- $z$ - partition function in Gibbs distribution
- $u(x)$ - energy function
- $V_c(x_c)$ for $c \in \mathcal{C}$ - potential function for clique $c$
- $X_s \in \Omega$ for $s \in S$ - discrete random field
- $X_s \in \{0, \cdots, M - 1\}$ for $s \in S$ - discrete random field
- $X_n \in \{0, \cdots, M - 1\}$ for $n \in \{0, \cdots, N - 1\}$ - discrete random field
MAP Segmentation (Chapter [16]) Notation:

- $Y_s \in \mathbb{R}^p$ for $n = 1, \cdots, N$ - observed data
- $X_s \in \Omega$ with $|\Omega| = \{0, \cdots, M - 1\}$ - unobserved class label
- $s \in S$ - pixel index space
- $l(y_s|x_s) = - \log p_{y_s|x_s}(y_s|x_s)$ - negative log likelihood ratio for each pixel
- $\theta_m = \{\pi_{m,j}, \mu_{m,j}, B_{m,j}\}_{j=0}^{J-1}$ - parameters of Gaussian mixture
- $\mathcal{T}$ - temperature in Gibbs distribution
Modeling Poisson Data (Chapter 17) Notation:

- \( Y_i \sim Pois(A_{i,s}x) \) for \( i = 1, \cdots, N \) - observed data
- \( X_j \) for \( j = 1, \cdots, N \) - unobserved rates
- \( l(x) = -\log p_{y|x}(y|x) \) - negative log likelihood ratio
Advanced Models (Chapter ??) Notation:
Bibliography


