Contents

Chapter 1

Image Restoration Using GMRFs

- Gradient is a column vector!

After so much theory on image modeling, you might be tempted to ask what this is good for? In this chapter, we begin to study applications of the image models we have introduced to problems such as image restoration. Our initial focus will be the use of the GMRF as a prior model for the problem of image restoration. This is a very nice example because the tractable structure of the GMRF model will allow us to obtain quite a bit of insight into both the value of model-based methods and the subtleties of the associated optimization algorithms typically used to compute their solutions. 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.

1.1 An Estimation Framework for Image Restoration

Let $X \in \Re^N$ be an N pixel image that we would like to measure, and let $Y \in \Re^N$ be the noisy measurements given by

$$Y = AX + W \tag{1.1}$$

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

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 Toeplize-block-Toeplitz matrix. In fact, with a little generalization, the model of (1.1) 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 (1.1), the conditional distribution of Y given X is

$$p_{y|x}(y|x) \ = \ \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left\{-\frac{1}{2\sigma^2}||y - Ax||^2\right\} \ . \tag{1.2}$$

One approach to recovering X from Y is to use the ML estimate of X given in Section **??**.

$$\begin{aligned}
\hat{x}_{ML} &= \arg\max_{x \in \Re^N} \log p_{Y|X}(y|x) \\
&= \arg\max_{x \in \Re^N} \left\{-\frac{1}{2\sigma^2}||y - Ax||^2 - \frac{N}{2}\log(2\pi\sigma^2)\right\} \\
&= \arg\min_{x \in \Re^N} ||y - Ax||^2 \\
&= A^{-1}y
\end{aligned}$$

However, this ML solution has a number of problems associated with it. First, if A has very small eigenvalues, the inversion of A maybe unstable. This can happen if the blurring filter's transfer function is nearly zero 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 can often produce a very noisy result. However, even if $A = I$, the identity matrix, the ML estimate is unsatisfying because it implies that the best restortion 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 is to use a Bayesian estimator from Section **??**. 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 will 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 σ_x^2. This results in a density function for X of the form

$$p_x(x) = \frac{1}{(2\pi)^{N/2}}|B|^{1/2} \exp\left\{-\frac{1}{2}x^t B x\right\} \tag{1.3}$$

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

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 tact, 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 multiplicative constant, but we can compute that constant because we know that $p_{x|y}(x|y)$ must integrate to 1. So using the expressions of (1.2) and (1.3), we have that

$$\log p_{x|y}(x|y) = -\frac{1}{2\sigma^2}||y - Ax||^2 - \frac{1}{2}x^t B x + c(y) \tag{1.4}$$

where $c(y)$ is some yet unknown function of y. Since (1.4) is a quadratic function of x, we can complete-the-square and express this function in the standard quadratic form

$$\log p_{x|y}(x|y) = -\frac{1}{2}(x - \mu(y))^t R^{-1}(x - \mu(y)) + c'(y) \tag{1.5}$$

where $\mu(y)$ and R are given by

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

and $c'(y)$ is only a function of y. From the form of (1.5), 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

$$p_{x|y}(x|y) = \frac{1}{(2\pi)^{N/2}}|R|^{-1/2}\exp\left\{-\frac{1}{2}(x-\mu(y))^t R^{-1}(x-\mu(y))\right\} \ . \qquad (1.6)$$

Using the posterior distribution of (1.6), we can compute some typical Bayesian estimators. Notice that the MMSE estimator is given by the conditional mean,

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

and the MAP estimate is given by the conditional mode

$$\begin{aligned} \hat{x}_{MAP} &= \arg\max_{x\in\Re^N} \log p_{X|Y}(x|y) \\ &= \arg\min_{x\in\Re^N} \left\{\frac{1}{2}(x-\mu(y))^t R^{-1}(x-\mu(y))\right\} \\ &= \mu(y) \ . \end{aligned}$$

So 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 of image features such as edges.

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

$$\hat{x} = \left(A^t A + \sigma^2 B\right)^{-1} A^t y \ . \qquad (1.7)$$

However, while the solution of (1.7) 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 each element is stored with a 64 bit float, then the matrix requires approximately 8 terra bytes of storage! Clearly we will need another approach.

1.2 Optimization for Computing the MAP Estimate

In order to make the computation of the MAP estimate more tractable, we need to go back to the first principles of its computation. The MAP estimate

of X given Y is given by

$$\begin{aligned}\hat{x}_{MAP} &= \arg\max_{x\in\Re^N} p_{x|y}(x|y) \\ &= \arg\max_{x\in\Re^N} \left\{\log p_{y|x}(y|x) + \log p_x(x) - \log p_y(y)\right\} \\ &= \arg\min_{x\in\Re^N} \left\{-\log p_{y|x}(y|x) - \log p_x(x)\right\} \ . \end{aligned} \tag{1.8}$$

Substituting the expressions from (1.2) and (1.3) into (1.8), yields the equations

$$\hat{x}_{MAP} = \arg\min_{x\in\Re^N} \left\{\frac{1}{2\sigma^2}||y - Ax||^2 + \frac{1}{2}x^t Bx\right\} \tag{1.9}$$

From equation (1.9), we see that, in this case, MAP estimation is equivalent to optimization of a quadratic function of the form

$$f(x) = \frac{1}{2\sigma^2}||y - Ax||^2 + \frac{1}{2}x^t Bx$$

The key insight is that while the direct inversion of (1.7) provides an exact mathematical solution, it requires an enormous amount of computation. Alternatively, numerical optimization methods can provide an approximate solution to the optimization of (1.9) that can be computed much less expensively, but at the cost of numerical approximation.

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

1.3 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 of minimizing the function. This approach to optimization is generally referred to as **gradient descent** optimization or **Jacobi** iterations.

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 β is a user selected step size. For the image restoration problem of (1.9), the gradient of $f(x)$ is given by

$$\nabla f(x) = -\frac{1}{\sigma^2} A^t(y - Ax) + Bx \; . \tag{1.10}$$

So for this case, the gradient descent update equation is given by

$$x^{(k+1)} = x^{(k)} + \alpha[A^t(y - Ax^{(k)}) - \sigma^2 Bx^{(k)}] \tag{1.11}$$

where $\alpha = \beta/\sigma^2$ is a scaled version of the step size. Rearranging terms, we have the recursion

$$x^{(k+1)} = (I - \alpha[A^tA + \sigma^2 B])x^{(k)} + \alpha A^t y \; . \tag{1.12}$$

An important special case of (1.12) 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 (1.11) has the form

$$x_s^{(k+1)} = x_s^{(k)} + \alpha[a_{-s} * (y_s - a_s * x_s^{(k)}) + \lambda(\delta_s - g_s) * x_s^{(k)}] \; , \tag{1.13}$$

where g_s is the non-causal predictor for the GMRF and $\lambda = \sigma^2/\sigma_x^2$ is a measure of 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.

1.3.1 Convergence Analysis of Gradient Descent

Gradient descent is a simple algorithm, but a key disadvantage is related to the difficulty in selection of the step-size parameter. Fig. 1.1 shows an

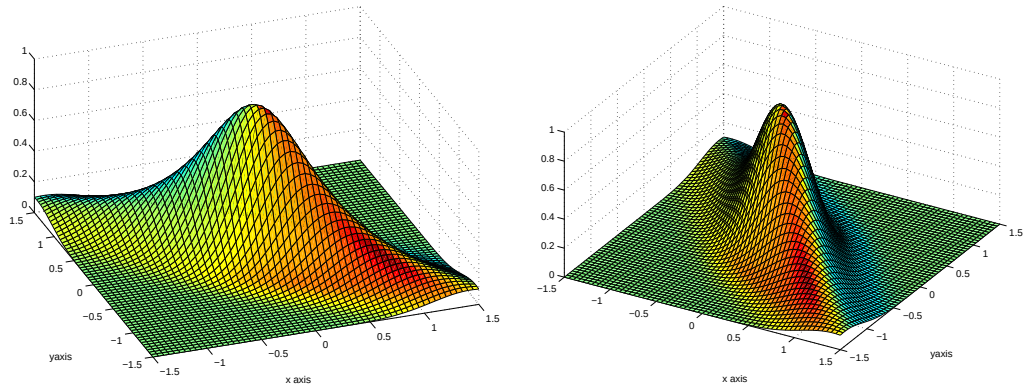
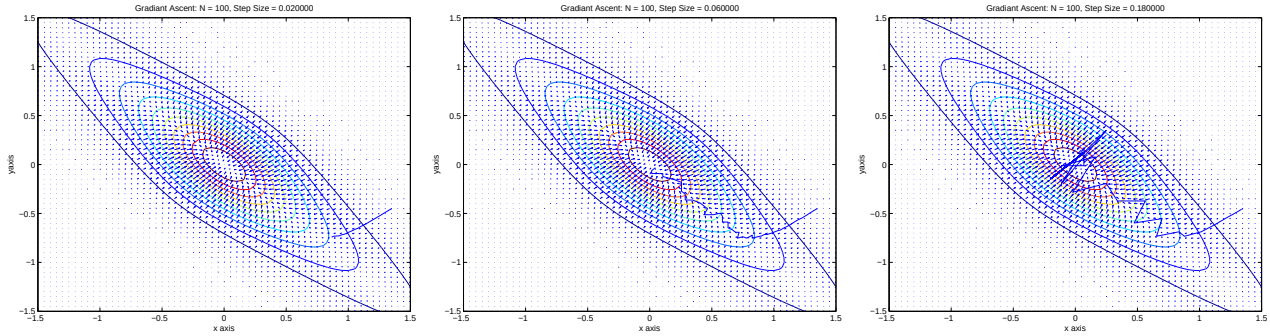


Figure 1.1: 3D rendering of the example non-quadratic cost function from two different views. The analytical form of the function is $\frac{1}{(x-y)^2+1}\exp(-4(x+y)^2)$.



(a) (b) (c)

Figure 1.2: 2D 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 α is slow, and the convergence with large α is unstable.

example of a 2-D non-quadratic function to be maximized, which is equivalent to the minimization of the negative function. Notice that the peak of the function is narrow along one dimension and wide along another.

Figure 1.2 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 Fig. 1.2(c), whereas too small of a step size results in the slow convergence of Fig. 1.2(a). Unfortunately, for high dimensional optimization problems such as (1.9), we will see that there is often no good choice of α.

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 (1.12) 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 (1.12) results in

$$x^{(\infty)} = (I - \alpha[A^t A + \sigma^2 B])x^{(\infty)} + \alpha A^t y \ . \tag{1.14}$$

Subtracting (1.12) and (1.14) results in

$$\epsilon^{(k+1)} = \left(I - \alpha[A^t A + \sigma^2 B]\right) \epsilon^{(k)} \ . \tag{1.15}$$

We can further simplify the structure of (1.15) by using eigenvector analysis of the matrix

$$H \stackrel{\triangle}{=} A^t A + \sigma^2 B \ .$$

Since H is symmetric, we know it can be diagonalized as $H = E\Lambda E^t$ where columns of E are the 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.

$$\tilde{\epsilon}^{(k)} = (I - \alpha\Lambda)^k \, \tilde{\epsilon}^{(0)} \ . \tag{1.16}$$

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

$$\tilde{\epsilon}_i^{(k)} = (1 - \alpha h_i)^k \, \tilde{\epsilon}_i^{(0)} \ . \tag{1.17}$$

Equation (1.17) 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 α so that convergence is fast for the largest eigenvalue, $h_{max} = \max_i\{h_i\}$. To do this we select α so that $1 - \alpha h_{max} = 0$, which implies that

$$\alpha = \frac{1}{h_{max}} \ .$$

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

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

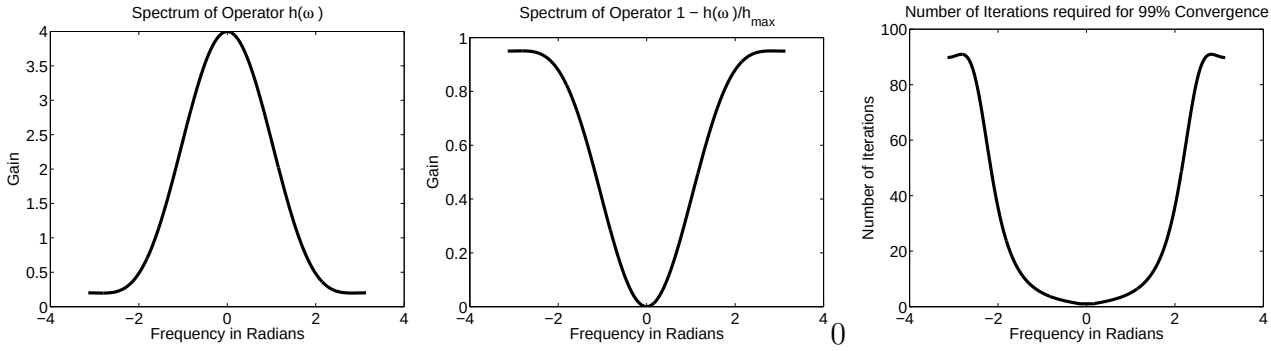
where $\kappa(H) = \frac{h_{max}}{h_{min}}$ is known as the **condition number** of the matrix H.[1] So if κ is very large, then $1 - \frac{1}{\kappa(H)}$ is very close to 1, and the convergence for that mode will be very slow.

In many applications, the condition number of H can be very large. When this is the case, it is not possible to choose α 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 α, then the modes with large eigenvalues can become unstable.

1.3.2 Frequency Analysis of Gradient Descent

In many practical situations, the matrix A represents a linear space-invariant filter. In this case, a more complete convergence analysis is possible through

[1] In general, the condition number of a matrix is given by $\kappa(A) = \left|\frac{\sigma_{max}}{\sigma_{min}}\right|$ where σ_{max} and σ_{min} are the maximum and minimum singular values of the matrix A.



(a) (b) (c)

Figure 1.3: Plots showing (a) the frequency response of $h(\omega)$, (b) the resulting function $1 - h(\omega)/h_{max}$ 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.

the use of frequency domain analysis. Taking the DSFT of (1.13) and rearranging terms yields

$$x^{(k+1)}(\omega) = \left(1 - \alpha\left[|a(\omega)|^2 + \lambda(1 - g(\omega))\right]\right) x^{(k)}(\omega) + \alpha a^*(\omega) y(\omega) \ . \quad (1.18)$$

If we further define $\epsilon^{(k)}(\omega)$ to be the DSFT of $\epsilon_s^{(k)} = x_s^{(k)} - x_s^{\infty}$, then we can derived the following recursion for the error in the computation of the MAP estimate.

$$\epsilon^{(k+1)}(\omega) = \left(1 - \alpha\left[|a(\omega)|^2 + \lambda(1 - g(\omega))\right]\right) \epsilon^{(k)}(\omega) \quad (1.19)$$

Using the definition that

$$h(\omega) \stackrel{\triangle}{=} |a(\omega)|^2 + \lambda(1 - g(\omega)) \ ,$$

we have that

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

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

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

Example 1.1: Consider a 1-D signal deconvolution problem of the form

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

where

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

is a smoothing filter, and W_n are i.i.d. $\sim N(0, \sigma^2)$ noise. We will use a prior model for X of a GMRF with a non-causal prediction variance of σ_x^2 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 $\sigma_x^2 = 10$, then

$$\begin{aligned} a(\omega) &= 1 + \cos(\omega) \\ 1 - g(\omega) &= 1 - \cos(\omega) \end{aligned}$$

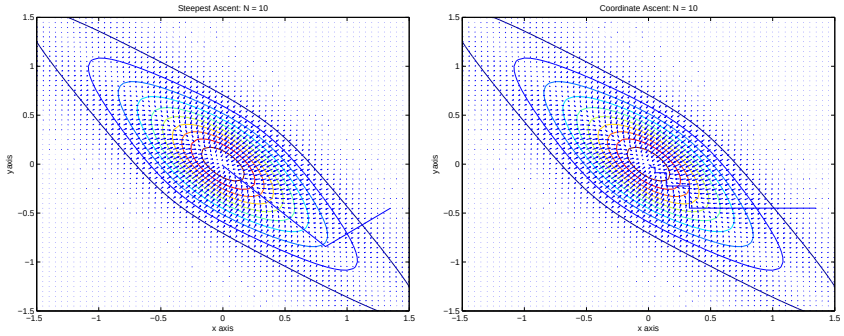
and we have that

$$h(\omega) \stackrel{\triangle}{=} |1 + \cos(\omega)|^2 + \frac{1}{10}\left(1 - \cos(\omega)\right) \ ,$$

Figure 1.3(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 $h_{max} = 4$ and $h_{min} = 1/5$. If we set the step size to $\alpha = 1/h_{max} = 1/4$, then the convergence at frequency ω is given by

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

Figure 1.3(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 1.3(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.



(a) (b)

Figure 1.4: 2D contour plots of the trajectory of a) steepest descent optimization and b) iterative coordinate descent optimization. Neither of these algorithms require the choice of a step parameter α.

This result 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 and color tend 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.

1.4 Steepest Descent Optimization

One partial solution to the problem of selecting a step size is to choose the value of α at each iteration that minimizes the cost function. This algorithm is called **steepest descent**.

The general form of the steepest descent optimization is given by

$$d^{(k)} = -\nabla f(x^{(k)}) \tag{1.20}$$

$$\alpha^{(k)} = \arg\min_{\alpha \geq 0} f(x^{(k)} + \alpha d^{(k)}) \tag{1.21}$$

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)} \ . \tag{1.22}$$

So in steepest descent optimization, each gradient step is taken with the step size that achieves the best minimum in that step. The computation of the step size in (1.22) requires a minimization of a 1D function and is known as a **line search**. Typically, it is possible to solve the line search by finding a

value of α which solves the following equation.

$$0 = \frac{\partial f(x^{(k)} + \alpha d^{(k)})}{\partial \alpha} = \left[\nabla f(x^{(k)} + \alpha d^{(k)})\right]^t d^{(k)} \tag{1.23}$$

In some cases, the line search required for steepest descent can be computationally intensive, but for the quadratic optimization it can be computed in closed form with modest computation. For the image restoration problem of (1.9), the step direction is given by

$$d^{(k)} \;=\; \frac{1}{\sigma^2} A^t (y - Ax^{(k)}) - Bx^{(k)} \;. \tag{1.24}$$

Using the gradient expression of (1.23) results in a closed form expression for the steep size of

$$\alpha^{(k)} = \sigma^2 \frac{||d^{(k)}||^2}{||d^{(k)}||_H^2} \tag{1.25}$$

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.

Figure 1.4(a) illustrates the results of steepest descent optimization for the cost function of Fig. 1.1. Notice that the convergence of steepest descent cost function is guarrented 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 to a limit.

1.5 Iterative Coordinate Descent Optimization

As we saw in Section 1.3.2, gradient descent tended to have slow convergence as high spatial frequencies. In practice, most gradient based methods tend to converge faster at low spatial frequencies for problem that require deblurring or deconvolution of an image. An alternative optimization approach with faster convergence at high spatial frequencies is the **iterative coordinate descent (ICD)** algorithm, which is also known as the *Gauss-Seidel*

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

$$x_s \leftarrow \arg\max_{x_s \in \Re} f(x)$$

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 $x + \alpha e_s$ where e_s is a vector that is 1 for element s, and 0 otherwise. Using this notation, we have that

$$x_s \leftarrow \arg\max_{\alpha \in \Re} f(x + \alpha e_s) \ .$$

The ICD update can then be computed by solving the equation

$$0 \ = \ \frac{\partial f(x + \alpha e_s)}{\partial \alpha} = \left[\nabla f(x + \alpha e_s)\right]^t e_s \ ,$$

which results in

$$\alpha = \frac{(y - Ax)^t A_{*,s} - \sigma^2 x^t B_{*,s}}{||A_{*,s}||^2 + \sigma^2 B_{s,s}} \ , \tag{1.26}$$

where $A_{*,s}$ and $B_{*,s}$ denote the s^{th} column of A and B respectively. This then results in the ICD update equation

$$x_s \leftarrow x_s + \frac{(y - Ax)^t A_{*,s} - \lambda \left(x_s - \sum_{r \in \partial s} g_{s-r} x_r\right)}{||A_{*,s}||^2 + \lambda} \ , \tag{1.27}$$

where $\lambda = \frac{\sigma^2}{\sigma_x^2}$ is the effective noise-to-signal ratio.

Direct implementation of the ICD algorithm using (1.27) requires the evaluation of the term $y - Ax$ with each pixel update. Typically, this requires too much computation. Fortunately, there is a simple method for speeding the computation by keeping a state variable to store this error term $e = y - Ax$. The pseudocode for this faster version of ICD is shown in Fig. 1.5.

As with gradient descent, an important special case of (1.27) 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

$$x_s \leftarrow x_s + \frac{a_{-s} * (y_s - a_s * x_s) - \lambda(x_s - \sum_{r \in \partial s} g_{s-r} x_r)}{\sum_s a_s^2 + \lambda} \ . \tag{1.28}$$

ICD Algorithm:

Initialize $e \leftarrow y - Ax$
For K iterations {
  For each pixel $i \in S$ {

$$\begin{aligned} v &\leftarrow x_i \\ x_s &\leftarrow x_s + \frac{e^t A_{*,s} - \lambda \left(x_s - \sum_{r \in \partial s} g_{s-r} x_r\right)}{||A_{*,s}||^2 + \lambda} \\ e &\leftarrow e - A_{*,s}(x_s - v) \end{aligned}$$

  }
}

Figure 1.5: Pseudocode for fast implementation of the ICD algorithm. The speedup depends on the use of a state variable to store the error, $e = y - Ax$.

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 λ is large and the signal-to-noise is low, then the update is weighted toward the neighboring pixel values; however, when λ is small and the signal-to-noise is high, then the update is weighted toward the measured data.

1.5.1 Convergence Analysis of Iterative Coordinate Descent

It is also possible to analyze the convergence of ICD, but the analysis is a bit more tricky [1]. We first define the matrix

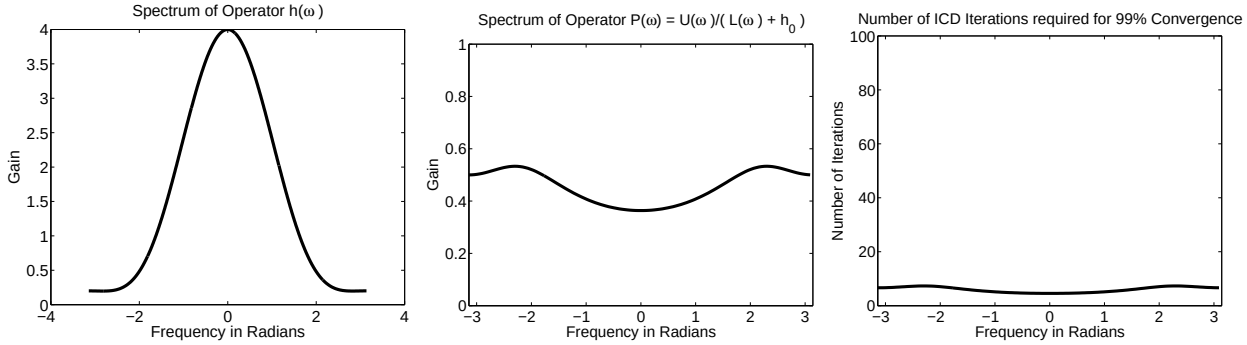
$$H \stackrel{\triangle}{=} A^t A + \sigma^2 B \ .$$

Using the gradient expression of (1.10), 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.

$$\left[\nabla f(x)\right]_s = \left[-A^t y + Hx\right]_s = 0$$



(a) (b) (c)

Figure 1.6: Plots showing (a) the frequency response of $h(\omega)$, (b) the resulting function $P(\omega) = \left|\frac{-U(\omega)}{L(\omega)+h_0}\right|$ which determines the convergence rate of ICD, (c) the number of iterations of gradient descent required for 99% convergence. Notice that ICD has more uniformly fast convergence across different frequencies.

This can be more explicitly written as

$$-\sum_{r\in S} a_{r,s}y_r + \sum_{r\in S} H_{s,r}x_r = 0 \ . \tag{1.29}$$

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 $x_r = x_r^{(k)}$, and the future pixels (i.e. $r > s$) still retain their old value, $x_r = x_r^{(k-1)}$. Substituting these values into (1.29) results in the following ICD update equation.

$$-\sum_{r\in S} a_{r,s}y_r + \left[\sum_{r<s} H_{s,r}x_r^{(k)}\right] + H_{s,s}x_s^{(k)} + \left[\sum_{r>s} H_{s,r}x_r^{(k-1)}\right] = 0 \ . \tag{1.30}$$

Notice that for $x_s = x_s^{(k)}$. This is because the solution of the equation results in the updated value of x_s.

Now since equation (1.30) 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 and

$$H = L + D + L^t \ .$$

Using this notation, equation (1.30) becomes

$$-A^t y + (L + D)x^{(k+1)} + L^t x^{(k)} = 0 \ .$$

So the update equation for $x^{(k)}$ becomes

$$x^{(k+1)} = -(L + D)^{-1}(L^t x^{(k)} - A^t y) \ .$$

If we define $\epsilon^{(k)} = x^{(k)} - x^{(\infty)}$, then this equation becomes

$$\epsilon^{(k+1)} = -(L + D)^{-1} L^t \epsilon^{(k)} \ . \tag{1.31}$$

The convergence of this equation can be studied in much the same way that we analyzed the convergence of equation (1.15) 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 $\frac{1}{\sigma_x^2}(\delta_s - g_s)$. So we have that multiplication by H is equivalent to convolution by h_s where

$$h_s = a_s * a_{-s} + \lambda(\delta_s - g_s) \ ,$$

where $\lambda = \sigma/\sigma_x^2$. 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

$$P(\omega) = \left| \frac{-L^*(\omega)}{L(\omega) + h_0} \right| \ . \tag{1.32}$$

If $P(\omega)$ is near zero, then convergence is rapid, but if it is near 1, then convergence is slow.

Example 1.2: Again consider the 1-D deconvolution example of the form

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

where

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

W_n are i.i.d. $\sim N(0, \sigma^2)$ noise, and X is a GMRF with a non-causal prediction variance of σ_x^2 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 $\sigma_x^2 = 10$, then

$$\begin{aligned} h_n &= a_n * a_{-n} + \frac{\sigma^2}{\sigma_x^2}(\delta_n - g_n) \\ &= \frac{3}{2}\delta_n + \left(\delta_{n-1} + \delta_{n+1}\right) + \frac{1}{4}\left(\delta_{n-2} + \delta_{n+2}\right) - \frac{1}{20}\left(\delta_{n-1} + \delta_{n+1}\right) \\ &= \frac{3}{2}\delta_n + \frac{19}{20}\left(\delta_{n-1} + \delta_{n+1}\right) + \frac{1}{4}\left(\delta_{n-2} + \delta_{n+2}\right) \ . \end{aligned}$$

So from this, we have that

$$l_s = \frac{19}{20}\delta_{n-1} + \frac{1}{4}\delta_{n-2} \ , \quad d_s = \frac{3}{2}\delta_n \ .$$

From this, we can calculate the function $P(\omega)$ from equation (1.32).

Figure 1.6(b) shows a plot of the function $P(\omega)$. Notice that its value is substantially less than 1 for all ω. This means the convergence will be relatively fast for a wide range of frequencies. Figure 1.3(c) makes this clearer by plotting the number of iterations required for 99% convergence of the ICD algorithm as a function of frequency. Notice that the number of iterations required for convergence is much less than that required for gradient descent, particularly at high spatial frequencies.

This result is typical of ICD which tends to converge rapidly at high spatial frequencies for this type of problem, but has slow convergence for very low spatial frequencies. When solving 2-D problems, this means that the high frequency detail, such as edges, converge quickly, but the low frequency shading and color offsets tend to converge more slowly.

1.6 Preconditioned Conjugate Gradient Optimization

Chapter 1 Problems

1. Derive the expressions for $\mu(y)$ and R used in equation (1.5).

2. 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".

3. Show that the update step for steepest descent is given by equation (1.25).

4. Show that the ICD update step is given by equation (1.26).

5. Consider the following functional

$$f(x) = ||y - x||^2 + x^t B x$$

where y and x are vectors, and B is symmetric and positive definite matrix.

a) Calculate the gradient decent algorithm for minimizing $f(x)$.

$$x^{(k+1)} = x^{(k)} + \omega \bigtriangledown_x f(x^{(k)})$$

b) Calculate the formulas of the ICD algorithm for minimizing $f(x)$.

c) Calculate the steepest decent algorithm for minimizing $f(x)$.

6. Let $X \in \Re^N$ be an N pixel image that we would like to measure, and let $Y \in \Re^N$ 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 $N(0, \sigma_w^2)$. Furthermore, in a Bayesian framework, assume that X is a GMRF with noncausal prediction filter g_s and noncausal prediction variance σ^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 a cost function $f(x)$ so that the MAP estimate of X is the minimum of this cost function.

7. Consider the optimization problem

$$\hat{x} = \arg\min_{x \in \Re^N} \left\{ ||y - Ax||^2 + \lambda x^t B x \right\}$$

where A is a nonsingular $N \times N$ matrix, B is a positive definite $N \times N$ matrix, and $\lambda > 0$.
a) Derive a closed form expression for the solution.
b) Calculate an expression for the gradient descent update using step size μ.
c) Calculate an expression for the steepest descent update.
d) Calculate an expression for the conjugate gradient update.
e) Calculate an expression for the coordinate descent update.

8. Consider the optimization problem

$$\hat{x} = \arg\min_{x \in \Re^N} \left\{ ||y - Ax||^2 + \lambda x^t B x \right\}$$

where A is a nonsingular $N \times N$ matrix, B is a symmetric positive definite $N \times N$ matrix, and $\lambda > 0$.
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?

Bibliography

[1] K. Sauer and C. A. Bouman. A local update strategy for iterative reconstruction from projections. *IEEE Trans. on Signal Processing*, 41(2):534–548, February 1993.