Statistical Learning Theory in Computational Neuroscience

J.G. Makin

July 14, 2022

Contents

Preface

This work in progress began as notes for—myself, not a class, and it shows; but it has since then steadily tended toward the format and style of a textbook. Some mistakes are almost guaranteed. If you stumble upon them and feel public-spirited, please do alert me at jg(my last name)`@purdue.edu`. The content is some combination of machine learning (graphical models, neural networks) and computational neuroscience, essentially dictated by what equations or algorithms I had, in my day job, to keep looking up or re-deriving, plus some glue to hold these together. At various later points in time, I was inspired to re-write in a more didactic style some section or other; these will be the most useful.

Large chunks of this collection were inspired by M.I. Jordan's fantastic, but still unpublished, textbook on statistical learning theory.

J.G.M.

Chapter 1

Introduction

Notation

This author firmly holds the (seemingly unpopular) view that good notation makes mathematical texts much easier to understand. More precisely, bad notation is much easier to parse—indeed, unremarkable—when one has already mastered the concepts; it can also mask deep underlying conceptual issues. I have attempted, although not everywhere with success, to use good notation in what follows.

Basic symbols. Basic notational conventions are for the most part standard. This book uses capital letters for random variables, lowercase for their instantiations, boldface italic font for vectors, and italic for scalar variables. The (generally Latin) letters for matrices are capitalized and bolded, but (unless random) in Roman font, and not necessarily from the front of the alphabet.

symbol	use
X, Y, Z	scalar random variables
x, y, z	scalar instantiations
$\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$	vector random variables
$\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$	vector instantiations
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{P}$, etc.	matrix
$\boldsymbol{\theta}, \boldsymbol{\phi}$	(non-random) parameters
$\boldsymbol{\pi}$	vector of categorical probabilities ($\sum_k \pi_k = 1$)
$\boldsymbol{\mu}$	mean (vector)
$\boldsymbol{\Sigma}$	covariance matrix

The set of all standard parameters (like means, variances, and the like) of a distribution are generally denoted as a single vector with either $\boldsymbol{\theta}$ or $\boldsymbol{\phi}$ (or, in a pinch, some nearby Greek letter). But note well that in the context of Bayesian statistics their status as random variables is marked in the notation: $\boldsymbol{\Theta}$, $\boldsymbol{\Phi}$. The Greek letters $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$ are generally reserved for particular parameters: the vector of categorical probabilities, the mean (vector), and the covariance matrix, respectively. Note that we do *not* use π for the transcendental constant; we use $\tau = 2\pi$ [6]

Operators. The symbol $\mathrm{Cov}[\cdot]$ is used with a single argument to denote the operator that turns a random variable into a covariance *matrix*; but with two arguments, $\mathrm{Cov}[\cdot, \cdot]$, to indicate the (cross) covariance between two random variables. Perhaps more idiosyncratically, angle brackets, $\langle \cdot \rangle$, are usually reserved for sample averages, as opposed to expectation

symbol	use
$\mathbb{E}[\boldsymbol{X}]$	expectation of $\boldsymbol{X}$
$\langle \boldsymbol{X} \rangle$	sample average of $\boldsymbol{X}$
$\mathrm{Var}[X]$	variance of X
$\mathrm{Cov}[\boldsymbol{X}]$	covariance matrix of $\boldsymbol{X}$
$\mathrm{Cov}[\boldsymbol{X}, \boldsymbol{Y}]$	covariance between $\boldsymbol{X}$ and $\boldsymbol{Y}$

values, although occasionally this stricture is relaxed.

Probability mass and density functions. We come now to a notational convention that, although widely employed in computer-science texts (and beyond), is particular dubious and warrants some explanation. In a statistics textbook, the probability-mass function associated with a discrete random variable X is usually written p_X or (to emphasize that it is a function) $p_X(\cdot)$, and the probability of a particular value x correspondingly as $p_X(x)$. To repeat, p_X and $p_X(\cdot)$ are functions, whereas $p_X(x)$ is a value. The subscript distinguishes this mass function from, say, one associated with the random variable Y, namely p_Y. Conditional distribributions, in turn, are written $p_{Y|X}$, and the value of a conditional distribution $p_{Y|X}(y|x)$.

Of course the domain of the conditional distribribution just exhibited is two-dimensional. With only a modicum of additional clutter, we can write the *unary* function that describes the distribution over Y for a particular setting x of X, $p_{Y|X}(\cdot|x)$. Or again, one can describe the probability of a particular y as a unary function of the conditioning variable, $p_{Y|X}(y|\cdot)$. These expressions are also fairly compact, but notice that there is really no way to write either without the raised dots (or some other artifice).

As the dimension of the domain of these functions increases, the notation becomes correspondingly unwieldy and ultimately impossible to parse. A few examples will make the point. Consider the joint distribution for a hidden Markov model, Fig. 2.5, in which the states on the backbone are latent but the "emissions" have been observed. Under the standard notation, this partially evaluated function would have to be written

$$p_{\boldsymbol{X}_1,\ldots,\boldsymbol{X}_T,\boldsymbol{Y}_1,\ldots,\boldsymbol{Y}_T}(\cdot,\ldots,\cdot,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T), \tag{1.1}$$

which is not attractive. The correspondence between some $\boldsymbol{X}_t$ and one of the raised dots is conceptually, but not visually, clear.

The problem is even more acute in the context of equations, where it collides with another (orthodox) notational convention. Consider this instance of Bayes's theorem (rule, formula, whatever) as it would typically be expressed by a statistician:

$$p_{X|Y,Z}(x|y,z) = \frac{p_{Y|X,Z}(y|x,z)p_{X|Z}(x|z)}{p_{Y|Z}(y|z)}. \tag{1.2}$$

Here one is making a statement not just about the left- and right-hand sides at some particular point (x, y, z), but for all such points in the range of X, Y, and Z, and therefore about ternary functions rather than some particular real numbers. In fine, the $\forall$ statements have been omitted.

This conventional omission is unproblematic unless one wants to talk about, say, the *binary* function $p_{X|Y,Z}(\cdot|\cdot,z)$. Now, one could simply write Eq. 1.2 with the appendage $\forall x, \forall y$, but that makes the omission in the generic case more appalling still. Furthermore, equations containing distributions like that in (1.1) would have to be littered with $\forall$ statements. The obvious alternative is to revert once again to the notation of raised dots. But there is a good reason why this notation is not used in equations. Bayes's theorem for $p_{X|Y,Z}(\cdot|\cdot,z)$ would come out as

$$p_{X|Y,Z}(\cdot|\cdot,z) = \frac{p_{Y|X,Z}(\cdot|\cdot,z)p_{X|Z}(\cdot|z)}{p_{Y|Z}(\cdot|z)}. \tag{1.3}$$

It is still possible to infer which omitted arguments on the left correspond to which on the right, but only by consulting the subscripts; the dots are just noise.

If the reader is not convinced of the obstacles this notation poses to rapid digestion, he is asked to compare

$$p_{\boldsymbol{X}_1,\ldots,\boldsymbol{X}_T,\boldsymbol{Y}_1,\ldots,\boldsymbol{Y}_T}(\cdot,\cdot,\ldots,\cdot,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T) = \prod_{t=1}^{T} p_{\boldsymbol{X}_t|\boldsymbol{X}_{t-1}}(\cdot|\cdot)p_{\boldsymbol{Y}_t|\boldsymbol{X}_t}(\boldsymbol{y}_t|\cdot) \tag{1.4}$$

with

$$p(\boldsymbol{X}_1,\ldots,\boldsymbol{X}_T,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T) = \prod_{t=1}^{T} p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1})p(\boldsymbol{y}_t|\boldsymbol{X}_t). \tag{1.5}$$

Both equations are intended to describe the joint distribution defined by a hidden Markov model (the left-hand side, as in Eq. 1.1 above), and its factorization (the right-hand side), conditioned on observations $\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T$. Eq. 1.5 employs a notation that is common in the literature of machine learning and could be interpreted, at the grossest level, merely as *defining* expressions like its left-hand side to be equal to the left-hand side of Eq. 1.4. Essentially, in the orthodox notation, the subscripts corresponding to observed variables are redundant, and the argument positions for the unobserved variables are filled with useless dots. Moving all subscript information into the arguments solves both problems at once, while maintaining the distinction between unevaluated and evaluated arguments with the standard notational distinction between random variables and their instantiations. But the proposal is not without difficulties.

First among these, $p(\boldsymbol{X})$ already has a meaning: it is the random variable generated by passing the random variable $\boldsymbol{X}$ through a particular function, namely $p(\cdot)$. This particular random variable shows up, for example, in the definition of information entropy, $\mathrm{H}[\boldsymbol{X}] = \mathbb{E}[-\log p(\boldsymbol{X})]$. How can this be reconciled with the use of $p(\boldsymbol{X})$ to indicate a(n unevaluated) function?

One recalls that a random variable, notoriously neither random nor a variable, is in fact a function. So far so good. But the domain of this function is the space of outcomes, not real numbers. In particular, unpacking the standard shorthand,

$$\begin{aligned} p(\boldsymbol{x}) &= \Pr[\{\omega \in \Omega | \boldsymbol{X}(\omega) = \boldsymbol{x}\}] \\ \implies p(\boldsymbol{X}(\hat{\omega})) &= \Pr[\{\omega \in \Omega | \boldsymbol{X}(\omega) = \boldsymbol{X}(\hat{\omega})\}]. \end{aligned}$$

To emphasize the difference between the two, consider the outcome space for a toss of two fair coins:

$$\hat{\omega} \in \Big\{ \text{🪙🪙}, \text{🪙🐇}, \text{🐇🪙}, \text{🐇🐇} \Big\} = \Omega.$$

Let $\boldsymbol{X}$ be the number of heads, in which case

$$\boldsymbol{x} \in \{0, 1, 2\}.$$

Then if $p(\boldsymbol{X})$ is interpreted as a probability "table," i.e., an unevaluated function, it is a table with four entries (which do not even sum to 1), whereas p has just three.

On the other hand, often when working with random variables, the outcome space is not explicitly specified, and we are free to interpret it as we please. For instance, in the present example we could imagine the outcome space to be

$$\Big\{ \text{no } \text{🪙}, \text{one } \text{🪙}, \text{two } \text{🪙} \Big\} = \Omega.$$

Then the domain of $p(\boldsymbol{X})$ is admittedly not identical, but at least isomorphic, to that of p. Such isomorphic outcome spaces will always be available. Thus, almost everything that we want to be able to say about the function p we can also say about the random variable $p(\boldsymbol{X})$.

In this text, we shall want to talk about partially evaluated probability-mass and probability-density functions precisely when evaluating at observations. Therefore, the appearance of lowercase letters as arguments to these functions will always indicate both that the values have been observed and that the function is being evaluated at those observations. The appearance of capital letters as the arguments will always indicate both that the variables have *not* been observed and that the corresponding arguments are still free; or, if one prefers, that the expression is in virtue of these random-variable arguments now a function of an outcome space isomorphic to the range of the those random variables. The raised-dot notation will not be used and subscripts to the probability functions altogether omitted.

Symbols for probability mass and density. This text indiscriminately uses the same letters for probability-mass and probability-density functions, in both cases the usual (for mass functions) p. Further semantic content is, however, communicated by diacritics. In particular, p is reserved for "the data distribution," i.e., the true source in the world of our samples, as opposed to a model. Often in the literature, but not in this book, the data distribution is taken to be a discrete set of points corresponding to a particular sample, that is, a collection of delta functions. Here, p is interpreted to be a full-fledged distribution, known not in form but only through the samples that we have observed from it.

symbol	**use**
p	the data mass/density function
$\hat{p}$, $\check{p}$	the model mass/density functions

For the model distribution we generally employ $\hat{p}$, although we shall also have occasion to use $\check{p}$ for certain model distributions.

Now, it is a fact from elementary probability theory [**?**] that a random variable carries with it a probability distribution. Conversely, it makes no sense to talk about two different probability distributions over the same random variable—although texts on machine learning routinely do, usually in the context of relative or cross entropy [**?**]. We will indeed often be interested in (e.g.) the relative entropy (KL divergence) of two distributions, p and $\hat{p}$, but this text takes pains to note that these are distributions over different random variables, for example $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$, respectively. In general, the text marks random variables and their corresponding distributions with the same diacritcis.

It may at first blush seem surprising, then, to see the KL divergence expressed as

$$\mathrm{D}_{\mathrm{KL}}\{p(\boldsymbol{Y})||\hat{p}(\boldsymbol{Y};\boldsymbol{\theta})\},$$

that is with $\boldsymbol{Y}$ on both sides. But indeed both distributions are being evaluated at $\boldsymbol{Y}$, and averaged under p. Still, doesn't the notational convention introduced in the previous section assert that the appearance of $\hat{\boldsymbol{Y}}$ in $\hat{p}\left(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\right)$ merely indicates that the second argument is free, in which case $\hat{p}(\boldsymbol{Y};\boldsymbol{\theta})$ is nonsensical? No: $\hat{p}(\boldsymbol{Y};\boldsymbol{\theta})$ is also a random variable, and *which* random variable it is depends on which random variable ($\boldsymbol{Y}$ or $\hat{\boldsymbol{Y}}$ or etc.) is inside the parentheses.

Similarly, although no difficulties are posed by the iterated expectation under $p(\boldsymbol{X}\,|\,\boldsymbol{Y})$

and $p(\boldsymbol{Y})$:

$$\mathbb{E}_{\boldsymbol{Y}}\left[\mathbb{E}_{\boldsymbol{X}|\boldsymbol{Y}}[f(\boldsymbol{X},\boldsymbol{Y})|\boldsymbol{Y}]\right] = \int_{\boldsymbol{y}} p(\boldsymbol{y}) \int_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y}) f(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y},$$

the iterated expectation under (first) the model distribution $\hat{p}\left(\hat{\boldsymbol{X}}\,\middle|\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\right)$ and (then) the *data* marginal, $p(\boldsymbol{Y})$ is perhaps less perspicuous in the notation:

$$\mathbb{E}_{\boldsymbol{Y}}\left[\mathbb{E}_{\hat{\boldsymbol{X}}|\boldsymbol{Y}}\left[f(\hat{\boldsymbol{X}},\boldsymbol{Y})\,\middle|\,\boldsymbol{Y}\right]\right] = \int_{\boldsymbol{y}} p(\boldsymbol{y}) \int_{\hat{\boldsymbol{x}}} \hat{p}(\hat{\boldsymbol{x}}|\boldsymbol{y};\boldsymbol{\theta}) f(\hat{\boldsymbol{x}},\boldsymbol{y})\,\mathrm{d}\hat{\boldsymbol{x}}\,\mathrm{d}\boldsymbol{y}.$$

The convention here is that the conditional distribution that is to appear on the right-hand side—in this case, a model distribution—is determined by the subscript to the (conditional) expectation symbol, by matching diacritics (or lack thereof). Thus in this case, the subscript $\hat{\boldsymbol{X}}$ indicates that a model distribution is required. The subscript vertical bar of course indicates conditioning. Whereas the capital letters *behind* the vertical bar, for their part, indicate what random variables this model distribution is to be conditioned on—in this case, $\boldsymbol{Y}$.

Derivatives [[The use of transposes in vector (and matrix) derivatives. The total derivative vs. partial derivatives. The "gradient" and the Hessian.]]

Part I

Representation and Inference

[[We usually begin our investigations with a set of variables and (assumed) statistical dependencies among them. Note that this set may include "latent" variables, of which we have made no observations. Inference, one of the two major operations at the center of this book (the other being *learning*), consists of estimating the unobserved or latent variables, using a model of some sort and the observations of the non-latent—"observed" or perhaps "patent"—variables. More precisely, one infers a probability distribution over some or all of these variables, conditioned on the observed variables.

In contrast, *learning* consists of finding the optimal numerical values for parameters of the model. Thus *inference* and *learning* are, respectively, questions in the domains of *probability* and *statistics*. Having said that, learning can be assimilated to inference (Bayesian inference), and inference can be assimilated to learning (variational inference).

In the next two chapters, we'll assume that our models of the data are "perfect"—although we'll still maintain the distinction between model and true generative process by using respectively p and $\hat{p}$ for their corresponding probability distributions.]]

Chapter 2

Directed Generative models

Generative vs. discriminative models. A *generative model* specifies a joint distribution over all random variables of interest. Now, what counts as a random variable, as opposed to a parameter, can itself be a decision for the modeler—at least for Bayesians (non-frequentists); but for now we set aside this question. Instead we might wonder what circumstances could justify specifying *less than* the entire joint distribution. [[See discussion in [13].]] One such circumstance is the construction of maps, e.g., from variables $\boldsymbol{Y}$ to $\boldsymbol{X}$. Such maps *can* be constructed by considering only the conditional distribution, $p(\boldsymbol{X} \mid \boldsymbol{Y})$, and ignoring the marginal distribution of $\boldsymbol{Y}$. These are known as *discriminative models*. *generative models* *discriminative models*

Perhaps now the case for discriminative learning of maps seems, not just plausible, but overwhelming. When is it helpful to model the joint distribution, $p(\boldsymbol{X}, \boldsymbol{Y})$, in the construction of a map (function) from $\boldsymbol{Y}$ to $\boldsymbol{X}$? One clear candidate is when we have some idea of the generative process, and it runs in the other direction. That is, suppose that data were (causally) generated by drawing some $\boldsymbol{x}$ from $p(\boldsymbol{X})$, followed by drawing a $\boldsymbol{y}$ from $p(\boldsymbol{Y}|\boldsymbol{x})$. Then it seems reasonable to build a model with matching structure, $\hat{p}\Big(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big) = \hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big)\hat{p}\Big(\hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big)$. If we do want a map from $\boldsymbol{Y}$ to $\boldsymbol{X}$, we will need to apply Bayes's theorem, which converts a source distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big)$, and emission distribution, $\hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big)$, into a recognition distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big)$.[1]

We recall that *Bayes's "theorem"* is just a rearrangement of the definition of conditional probabilities: *Bayes's theorem*

$$\begin{aligned}
\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\boldsymbol{y}; \boldsymbol{\theta}\Big) &= \frac{\hat{p}\Big(\boldsymbol{y}\Big|\hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big)\hat{p}\Big(\hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big)}{\hat{p}(\boldsymbol{y}; \boldsymbol{\theta})} \\
&= \frac{\hat{p}\Big(\boldsymbol{y}\Big|\hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big)\hat{p}\Big(\hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big)}{\int_{\hat{\boldsymbol{x}}} \hat{p}(\boldsymbol{y}|\hat{\boldsymbol{x}}; \boldsymbol{\theta})\hat{p}(\hat{\boldsymbol{x}}; \boldsymbol{\theta})\,\mathrm{d}\hat{\boldsymbol{x}}} \\
&\propto \hat{p}\Big(\boldsymbol{y}\Big|\hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big)\hat{p}\Big(\hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big).
\end{aligned} \tag{2.1}$$

The last formulation is, although less explicit, also common.[2] The idea is to emphasize

[1] In this context, the source, emission, and recognition distributions are of course often referred to as the prior, likelihood, and posterior of $\hat{\boldsymbol{X}}$. But these terms are overloaded: e.g., the "maximum-likelihood" estimate refers to the likelihood of the parameters, $\boldsymbol{\theta}$, not of a random variable, $\hat{\boldsymbol{X}}$. Indeed, the term "likelihood" was introduced in a purely frequentist context by Fisher [4]. The term "recognition distribution" originates in [3].

[2] I have written Bayes's theorem with a lowercase $\boldsymbol{y}$, in order to emphasize that it provides a distribution

that the formula relates three functions of the same variable, $\hat{\boldsymbol{X}}$: what we have called the recognition, emission, and source distributions. Thus the "omitted" proportionality constant may depend on $\boldsymbol{y}$, but not on $\hat{\boldsymbol{X}}$. Indeed, for certain distributions, applying Bayes's theorem will not require computing the normalizer explicitly; instead, we shall simply recognize the parameteric family to which the unnormalized product of source and emission distributions (or prior and likelihood) belong. Won't we need the normalizer to make further computations with the recognition distribution? Not necessarily: for some distributions, the computations can be written purely in terms of the cumulants, which can be computed independently of the normalizer.

Then again, for other distributions, the normalizer is essential—for example, if $\hat{\boldsymbol{X}}$ were categorically distributed. And often it is useful in its own right: for some models, we *never* make observations of $\boldsymbol{X}$, only $\boldsymbol{Y}$, so $\hat{p}(\boldsymbol{y};\boldsymbol{\theta})$ provides the ultimate measure of our model's worth. Accordingly, in many of the models that we consider below, we shall compute it. That is, we shall convert the source and emission distributions into the recognition distribution (with Eq. 2.1) and this mariginal distribution over the observations. We will thereby have "inverted the arrow" of the model: provided an alternative, albeit completely equivalent, characterization of the joint distribution of $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$.

Unfortunately (and now we arrive at the rub), this is possible only for a handful of distributions. We shall explore this problem over the course of this chapter.

Where does the generative model come from? .... [[some examples]] We have discussed the possibility of constructing generative models from "some idea of the generative process," and in certain cases this includes even the numerical values of parameters; e.g., perhaps they come from a physical process. More frequently, we need to *learn* these parameters. This task will occupy other chapters in this book, but a basic distinction between learning tasks has implications for our representations themselves.

The distinction is whether or not we *ever* observe the variables about which, ultimately, we shall make inferences. In one kind of problem, we at some time observe the source variables along with the emissions, i.e. we make observations $\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N$ from the data distribution $p(\boldsymbol{X}, \boldsymbol{Y})$, and fit a model $\hat{p}\left(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}; \boldsymbol{\theta}\right)$ to these data. In the other kind of problem, we *never observe a variable* $\boldsymbol{X}$, and instead only ever observe $\{\boldsymbol{y}_n\}_{n=1}^N$ from the data marginal $p(\boldsymbol{Y})$. In this case there is no $\boldsymbol{X}$ to speak of, only $\hat{\boldsymbol{X}}$. At first blush, it may seem somewhat mysterious why we would introduce into our model a variable which may have no counterpart in the real world. But such "latent" variables can simplify our observations, as seen most obviously in a mixture model, Fig. **??**. Although we never directly observe the value of this latent variable, it seems obvious that it is there. Other examples include [[spatiotemporally extended objects for images...]]

What is the implication for our representations? Latent-variable models will in general be less expressive than their otherwise equivalent, fully observed counterparts. This is because only certain aspects of the latent variable will ever be identifiable from the observed data. For example, consider a normally distributed latent variable, $\hat{\boldsymbol{X}}$, and an emission $\hat{\boldsymbol{Y}}|\hat{\boldsymbol{X}}$ that is normally distributed about an affine function of $\hat{\boldsymbol{X}}$. If the offset in that affine function is unknown and to be learned, there is no point in allowing the latent variable a non-zero mean. It provides one degree of freedom too many. More generally, latent-variable

over $\hat{\boldsymbol{X}}$ for a *particular* observation (drawn from the data distribution, $p(\boldsymbol{Y})$). But this is of course true for any observation $\hat{\boldsymbol{y}}$, so the equation is equally valid for $\hat{\boldsymbol{Y}}$, i.e., as a statement about a function of two random variables rather than one.

models raise questions of the identifiability of their parameters. We discuss these issues below.

Directed graphical models. [[Of course, our model may involve more than just two random variables! Then it may become quite useful to express graphically the statistical dependencies among these variables. And indeed, when the dependencies are described, as here, in terms of probability distributions, we can use these distributions to parameterize a directed acyclic graph, each node corresponding to a random variable.

Now, by the chain rule of probability, the joint distribution can always be written as a product over N conditional distributions (with marginal distributions as a special case), one for each of the N variables in the joint. Thus a one-to-one-to-one relationship is established between nodes, random variables, and conditional distributions. The variable to the left of the vertical bar | therefore determines the assignment of conditional distribution to node—and the variables to the right of the bar, for their part, determine the parents of that node in the graph. That is, for all i, the conditional distribution $\hat{p}(\hat{X}_i|\hat{X}_{\mathrm{p}(i)})$ is assigned to node i, and nodes in the set $\mathrm{p}(i)$ are connected by directed edges (arrows) to the node i. (Nodes with only marginal distributions have no parents.)

directed graphical model

There are two problems with this approach. First, if the conditional distributions were in fact adduced simply by naïve application of the chain rule of probability, then one node in the graph would have all the others as parents, another would have nearly all, etc. However, the fundamental practical fact about graphical models, as we shall see, is that they are only really useful when this doesn't happen; indeed, when most of the arrows in the graph are missing. That raises the question: Given the semantics of these directed graphical models, when can we remove arrows? The second problem is that this graph doesn't capture any of the statistical dependencies particular to our model! The chain rule applies equally to any joint distribution.

The problems are flip sides of the same coin and have a single solution. When there are conditional independencies among variables, some of the conditional distributions simplify: variables disappear from the right-hand side of the vertical bars |. Given the rules for constructing the graph lately described, this corresponds to removing arrows from the graph. Thus, missing arrows in the graph represent (condititional) independence statements, and make inference possible, as we shall see.

Now, chain-rule decompositions are not unique, and so in practice it is unusual simply to write out the joint in terms of one of these and then to start thinking of what conditional independence statements apply. Instead, one typically proceeds from the other direction, by considering how certain random variables depend on others, constructing conditional distributions accordingly (and the graph along the way), and then finally multiplying them together to produce the joint.

Unsuprisingly, then, conditional (in)dependencies between *any* two (sets of) variables can be determined with a simple procedure on these directed graphical models. And (exact) inference amounts to some more or less clever application of Bayes's rule, Eq. 2.1, that exploits the graph structure. How easy or hard it is to apply the rule depends on both the structure of the graphical model and the nature of the distributions that parameterize it. [(1) Large cliques are bad. (2) Only certain distributions yield nice, closed-form recognition distributions. We can discretize the support, so a solution is always possible, but in fact this is the worst case..... The root problem is marginalizing out nuisance variables. .... This dependence can be quite restrictive in practice, which motivates the use of *approximate*

inference techniques....]

[[...]]

In the following sections, we consider models with independent and identically distributed (i.i.d.) Gaussian emissions. To generate different models, we consider source distributions that differ along two abstract "dimensions": (1) *sparsity*; and (2) internal structure of statistical dependence—in particular, we consider source variables that form a Markov chain.

Naming conventions for the models are not wholly satisfactory. Often the most popular name associated with a model arose historically for the inference algorithm, or even the corresponding *learning* algorithm, rather than the model itself. Where possible, which is not always, I have supplied a name with some basis in the literature that describes the model itself.

2.1 Static models

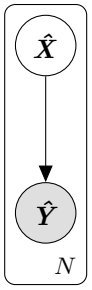
As promised, we consider models with independent and identically distributed (i.i.d.) Gaussian emissions, but with different distributions of the "source" variable. We start with the "sparsest," a mixture model.

2.1.1 Mixture models and the GMM

include a plot of example data

Consider the graphical model in Fig. 2.1. The generative process could be described as first rolling a K-sided die in order to pick a mean (vector) and covariance (matrix) from a set of means and covariances, followed by a draw from a multivariate normal distribution described by these parameters:

$$\begin{aligned} \hat{p}\Big(\hat{X};\boldsymbol{\theta}\Big) &= \text{Categ}(\boldsymbol{\pi}) \\ \hat{p}\Big(\hat{\boldsymbol{Y}}\,\Big|\,\hat{X};\boldsymbol{\theta}\Big) &= \mathcal{N}\Big(\boldsymbol{\mu}_{\hat{X}},\boldsymbol{\Sigma}_{\hat{X}}\Big). \end{aligned} \tag{2.2}$$



$$\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \text{Categ}(\boldsymbol{\pi})$$

$$\hat{p}\Big(\hat{\boldsymbol{Y}}\,\Big|\,\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \mathcal{N}\Big(\boldsymbol{\mu}_{\hat{X}},\boldsymbol{\Sigma}_{\hat{X}}\Big)$$

Figure 2.1: **Gaussian mixture model.**

Here, $\boldsymbol{\pi}$ is a vector of the probabilities of the K classes (sides of a possibly loaded die); its elements sum to one.

In Eq. 2.2, the support of $\hat{X}$ could be the integers between 1 and K (inclusive). However, a common alternative represention for the categorical random variable is a *one-hot vector*, $\hat{\boldsymbol{X}}$, i.e. a vector consisting of all zeros except for a single 1 at the category being represented. This allows us to rewrite the model in other convenient forms. For example, the source distribution can place the random variable into the exponent,

$$\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \prod_{k=1}^{K} {\pi_k}^{\hat{X}_k}, \tag{2.3}$$

which is particularly useful when working with log probabilities. The emission can be expressed similarly,

$$\hat{p}\Big(\hat{\boldsymbol{Y}}\,\Big|\,\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \prod_{k=1}^{K} \Big[\mathcal{N}\Big(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\Big)\Big]^{\hat{X}_k}; \tag{2.4}$$

or, alternatively, its mean can be expressed as a *linear* function of the die roll,

$$\hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \mathcal{N}\Big(\mathbf{C}\hat{\boldsymbol{X}}, \boldsymbol{\Sigma}_{\hat{\boldsymbol{X}}}\Big), \tag{2.5}$$

where the columns of the matrix $\mathbf{C} = [\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K]$ are the mean vectors. But this formulation cannot be extended elegantly to the covariance matrix, which is therefore the chief advantage of Eq. 2.4 over Eq. 2.5: the dependence on the latent variable of the covariance matrix, as well as of the mean, can be expressed algebraically. We shall see an example of its usefulness when we turn to learning in the Gaussian mixture model, Section 6.4.1.

Inference in the GMM. Inference in this model is a simple application of Bayes' rule. We start by noting that the recognition distribution is necessarily another categorical distribution (i.e., a toss of a die); the only question is what the probabilities of each category (each side of the die) are.

We begin somewhat pedantically with Bayes's theorem as it is expressed in Eq. 2.1, but for a discrete random variable represented one-hot. To emphasize that we are making inferences from actual observations, we again write this in terms of an observation from the data distribution, $\boldsymbol{y}$:

$$\begin{aligned}
\hat{p}\Big(\hat{X}_j = 1\Big|\boldsymbol{y};\boldsymbol{\theta}\Big) &= \frac{\hat{p}\Big(\boldsymbol{y}\Big|\hat{X}_j = 1;\boldsymbol{\theta}\Big)\hat{p}\Big(\hat{X}_j = 1;\boldsymbol{\theta}\Big)}{\sum_{\hat{\boldsymbol{x}}}\hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{x}};\boldsymbol{\theta}\Big)\hat{p}(\hat{\boldsymbol{x}};\boldsymbol{\theta})} \\
&= \frac{\hat{p}\Big(\boldsymbol{y}\Big|\hat{X}_j = 1;\boldsymbol{\theta}\Big)\hat{p}\Big(\hat{X}_j = 1;\boldsymbol{\theta}\Big)}{\sum_{k=1}^{K}\hat{p}\Big(\boldsymbol{y}\Big|\hat{X}_k = 1;\boldsymbol{\theta}\Big)\hat{p}\Big(\hat{X}_k = 1;\boldsymbol{\theta}\Big)} \\
&= \frac{\hat{p}\Big(\boldsymbol{y}\Big|\hat{X}_j = 1;\boldsymbol{\theta}\Big)\pi_j}{\sum_{k=1}^{K}\hat{p}\Big(\boldsymbol{y}\Big|\hat{X}_k = 1;\boldsymbol{\theta}\Big)\pi_k} \\
&= \frac{\exp\Big\{\log\Big(\hat{p}\Big(\boldsymbol{y}\Big|\hat{X}_j = 1;\boldsymbol{\theta}\Big)\pi_j\Big)\Big\}}{\sum_{k=1}^{K}\exp\Big\{\log\Big(\hat{p}\Big(\boldsymbol{y}\Big|\hat{X}_k = 1;\boldsymbol{\theta}\Big)\pi_k\Big)\Big\}} \\
&= \mathrm{softmax}\{\hat{\boldsymbol{z}}\}_j,
\end{aligned} \tag{2.6}$$

i.e., the j^{th} output of the *softmax function*, where $\hat{z}_k = \log\Big(\hat{p}\Big(\boldsymbol{y}\Big|\hat{X}_k = 1;\boldsymbol{\theta}\Big)\pi_k\Big)$. For the vector of all possible categories, $\hat{\boldsymbol{X}}$, the recognition distribution is therefore

softmax function, or more properly the *soft argmax*

$$\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\boldsymbol{y};\boldsymbol{\theta}\Big) = \mathrm{Categ}(\mathrm{softmax}\{\hat{\boldsymbol{z}}\}) \tag{2.7}$$

The derivation so far is general to any mixture model. For the GMM's Gaussian emissions, Eq. 2.4, $\hat{z}_k$ becomes

$$\hat{z}_k = c - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| - \frac{1}{2}\Big(\boldsymbol{y} - \boldsymbol{\mu}_k\Big)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}\Big(\boldsymbol{y} - \boldsymbol{\mu}_k\Big) + \log\pi_k, \tag{2.8}$$

where the constant c is irrelevant since it occurs in all the terms and therefore factors out of the softmax. This quantity, $\hat{z}_k$, is (up to an additive constant) the log of the recognition probability of class k. What does it mean?

One way to interpret Eq. 2.8 is (mentally) to fix $\hat{z}_k$ and see what the contours of constant (log-)probability look like. Or again, one can set $\hat{z}_k \stackrel{\text{set}}{=} \hat{z}_j$ for some $j \neq k$, in which case the resulting expression (in $\boldsymbol{y}$) is the boundary between classes k and j. In either case, these expressions are second-order polynomials in $\boldsymbol{y}$, a property inherited from the normal distribution. This is illustrated in Fig. **??**. If we sought fancier (higher-order) boundaries between classes, we would need a distribution with more "shape" parameters.

insert figure of quadratic class boundaries here; also a subfig of linear boundaries with the projection.....

Moving toward simpler, rather than more complex, contours, consider now the special case of constant covariance matrices across classes. Then the terms $-\frac{1}{2}\log|\boldsymbol{\Sigma}|$ and $-\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{y}$ could be factored out of all the elements of $\hat{\boldsymbol{z}}$, and cancelled (like c). That leaves

$$-\frac{1}{2}\Big(\boldsymbol{\mu}_k{}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - 2\boldsymbol{\mu}_k{}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{y}\Big) + \log\pi_k,$$

a *linear* function of $\boldsymbol{y}$. Setting two such expressions equal to each other for different classes k and j (and applying a little bit of algebra), we see that two classes are equiprobable when

$$\left(\boldsymbol{y} - \frac{1}{2}\Big(\boldsymbol{\mu}_k + \boldsymbol{\mu}_j\Big)\right)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\Big(\boldsymbol{\mu}_k - \boldsymbol{\mu}_j\Big) = \log\left(\frac{\pi_j}{\pi_k}\right). \tag{2.9}$$

Class k is more probable than class j when the left-hand side is larger than the right, and *vice versa*. (Notice, though, that in neither case is one of these classes guaranteed to be *most* likely, since we are here ignoring all other classes).

We can even ignore the covariance matrix if we are willing to work in the whitened space (i.e., absorb a factor of $\boldsymbol{\Sigma}^{-1/2}$ into $\boldsymbol{y}$ and $\boldsymbol{\mu}$). Then Eq. 2.9 becomes transparent. To decide the relative probability of a (whitened) observation belonging to class k or class j, we first measure how far it is from the midpoint between the class centers, $\frac{1}{2}\Big(\boldsymbol{\mu}_k + \boldsymbol{\mu}_j\Big)$. We then project this (whitened) displacement onto the vector connecting their (whitened) centers. Here the vector runs from j to k, so the probability of class k increases with (positive) distance along the projection. But the point of equality on the projection is not zero displacement, because one class may be *a priori* more probable than the other. The term on the right-hand side accounts for this.

Marginalization in mixture models. As noted at the outset, our evaluation of the model often requires $\hat{p}(\boldsymbol{y};\boldsymbol{\theta})$. It is easily read off the denominator in our derivation of a mixture model's recognition distribution:

$$\hat{p}(\boldsymbol{y};\boldsymbol{\theta}) = \sum_{k=1}^{K}\hat{p}\Big(\boldsymbol{y}\Big|\hat{X}_k = 1;\boldsymbol{\theta}\Big)\pi_k. \tag{2.10}$$

For any easily evaluable emission distributions, this is computed painlessly: the probability of the datum $\boldsymbol{y}$ under mixture component k, scaled by the prior probability of component k, and then summed over all k.

Mixture models and conjugacy. Comparing Eqs. 2.7 and 2.2, we see that the recognition distribution over $\hat{\boldsymbol{X}}$ is in the same family as the source distribution—to wit, the family of categorical distributions. We need not even have carried out any derivation to see this: the support of $\hat{\boldsymbol{X}}$ is the set of length-K one-hot vectors (or alternatively, the support of $\hat{X}$ is the set of integers from 1 to K), any distribution over which can be described as

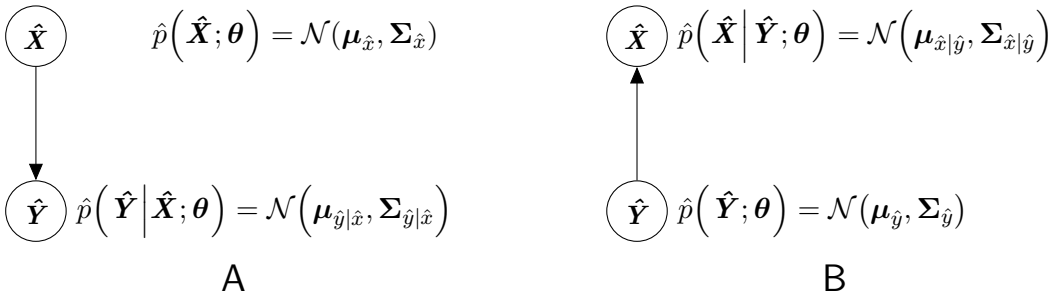


Figure 2.2: **Emission and source.** These models make equivalent independence statements about $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$, but parameterize the distribution differently.

categorical. Now recall the definition of a *conjugate prior*: the prior distribution over a parameter of the likelihood is conjugate to that likelihood when the posterior distribution over the parameter is in the same family as the prior. So we interpret the "source" variable $\hat{\boldsymbol{X}}$ as a set of parameters, albeit with some trepidation, since the parameters of exponential families are always continuous rather than discrete (although see [24]). Still, comparing Eqs. 2.3 and 2.4, we see that the "prior" distribution mimics the "likelihood" in the usual way, with the sufficient statistics being the vector of log probabilities. Thus encouraged, we put the emission in exponential-family form, *conjugate prior*

$$\hat{p}\Big(\boldsymbol{y}\Big|\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \exp\left\{\sum_{k=1}^{K} \hat{X}_k \log\Big(\hat{p}\Big(\boldsymbol{y}\Big|\hat{X}_k;\boldsymbol{\theta}\Big)\Big)\right\},$$

and the source distribution into the standard form for conjugate priors:

$$\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \exp\left\{\sum_{k=1}^{K} \hat{X}_k \log \pi_k\right\}.$$

The lack of a log-partition function should give us pause—it says that the partition function is unity for all values of the "parameters"—but we could nevertheless interpret $\log \boldsymbol{\pi}$ as the (first) standard hyperparameter of the conjugate prior. To compute the recognition (posterior) distribution, this gets updated in the standard way for conjugate priors, namely, by adding the sufficient statistics: $\log \pi_k + \log\Big(\hat{p}\Big(\boldsymbol{y}\Big|\hat{X}_k;\boldsymbol{\theta}\Big)\Big)$ for all k.

In fine, the categorical distribution is in some sense "conjugate" to any likelihood function of log probability. This is one way of understanding why mixture models are amenable to Bayesian analyses....

2.1.2 Jointly Gaussian models and factor analyzers

We begin with a more generic model than the ones which will be useful to us later. In particular, we begin simply by replacing the categorical latent variable from the previous section with a (vector) standard normal variate, as in Fig. 2.2A. We refine this slightly by assuming further that (1) the emission mean is an affine function of the source, $\hat{\boldsymbol{X}}$; and (2) the emission covariance is fixed for all values of the $\hat{\boldsymbol{X}}$:

$$\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \mathcal{N}(\boldsymbol{\mu}_{\hat{x}}, \boldsymbol{\Sigma}_{\hat{x}}), \qquad \hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \mathcal{N}\Big(\mathbf{C}\hat{\boldsymbol{X}} + \boldsymbol{c}, \boldsymbol{\Sigma}_{\hat{y}|\hat{x}}\Big). \tag{2.11}$$

Inference in jointly Gaussian models. To compute the recognition distribution, we of course use Bayes's theorem, Eq. 2.1, but this time we take advantage of a property of the source and emission distributions alluded to at the start of the chapter. As this example nicely illustrates, it will not be necessary to compute the normalizer explicity. That is because we will recognize the form of the distribution—in this case, another Gaussian—even without the normalizer. This property follows from the fact that both the source and the emission are exponentiated second-order polynomials in $\hat{\boldsymbol{X}}$. Multiplying two such functions together yields a *third* exponentiated second-order polynomial in $\hat{\boldsymbol{X}}$ (adding second-order polynomials can't increase their order), which must be a third normal distribution.

There are circumstances under which this is really all we need to know; but what if we want the normalizer explicity? We still don't have to take the integral, because the normalizer for a Gaussian can be expressed simply as a function[3] of the covariance matrix: $Z = \tau^{K/2}|\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}|^{1/2}$. (Another way to think about this is that someone had to take the integral once, but we don't have to do it again.) Naturally, to compute this we do need an expression for the covariance matrix, so it won't do to leave the second-order polynomial in any form whatever; we need to re-arrange it into a quadratic form, with the inverse covariance matrix in the middle. That is the only algebraic work that actually needs to be done in applying Bayes's theorem:

$$\begin{aligned}
\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) &\propto \hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) \\
&\propto \exp\left\{-\frac{1}{2}\Big(\mathbf{C}\hat{\boldsymbol{X}}+\boldsymbol{c}-\hat{\boldsymbol{Y}}\Big)^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1}\Big(\mathbf{C}\hat{\boldsymbol{X}}+\boldsymbol{c}-\hat{\boldsymbol{Y}}\Big)-\frac{1}{2}\Big(\hat{\boldsymbol{X}}-\boldsymbol{\mu}_{\hat{x}}\Big)^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{x}}^{-1}\Big(\hat{\boldsymbol{X}}-\boldsymbol{\mu}_{\hat{x}}\Big)\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\hat{\boldsymbol{X}}^{\mathrm{T}}\Big(\mathbf{C}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1}\mathbf{C}+\boldsymbol{\Sigma}_{\hat{x}}^{-1}\Big)\hat{\boldsymbol{X}}-2\Big(\Big(\hat{\boldsymbol{Y}}-\boldsymbol{c}\Big)^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1}\mathbf{C}+\boldsymbol{\mu}_{\hat{x}}{}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{x}}^{-1}\Big)\hat{\boldsymbol{X}}\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\Big[\hat{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}^{-1}\hat{\boldsymbol{X}}-2\boldsymbol{\mu}_{\hat{x}|\hat{y}}{}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}^{-1}\hat{\boldsymbol{X}}\Big]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\Big(\hat{\boldsymbol{X}}-\boldsymbol{\mu}_{\hat{x}|\hat{y}}\Big)^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}^{-1}\Big(\hat{\boldsymbol{X}}-\boldsymbol{\mu}_{\hat{x}|\hat{y}}\Big)\right\} \\
&= \mathcal{N}\Big(\boldsymbol{\mu}_{\hat{x}|\hat{y}},\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\Big),
\end{aligned} \tag{2.12}$$

where we have defined

$$\boldsymbol{\mu}_{\hat{x}|\hat{y}} =: \boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\Big[\mathbf{C}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1}(\,\hat{\boldsymbol{Y}}-\boldsymbol{c})+\boldsymbol{\Sigma}_{\hat{x}}^{-1}\boldsymbol{\mu}_{\hat{x}}\Big] \tag{2.13}$$

and

$$\boldsymbol{\Sigma}_{\hat{x}|\hat{y}} =: \Big(\mathbf{C}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1}\mathbf{C}+\boldsymbol{\Sigma}_{\hat{x}}^{-1}\Big)^{-1}. \tag{2.14}$$

These recognition cumulants should be somewhat intuitive. The recognition mean is a convex combination of the information from the emission and the source distributions. More precisely, it is a convex combination of the source mean, $\boldsymbol{\mu}_{\hat{x}}$, and the centered observation that has been transformed into the latent space, $\mathbf{C}^{\dagger}(\,\hat{\boldsymbol{Y}}-\boldsymbol{c})$. Here $\mathbf{C}^{\dagger}$ is a pseudo-inverse of $\mathbf{C}$.[4] The weights are the (normalized) precisions of these two distributions, although the precision of the emission about $\hat{\boldsymbol{X}}$ must be transformed into the latent space first with

[3] We use $\tau = 2\pi$ throughout this book [6].

[4] When $\hat{\boldsymbol{X}}$ is higher-dimensional than $\hat{\boldsymbol{Y}}$, this could be any right inverse. When $\hat{\boldsymbol{Y}}$ is higher-dimensional than $\hat{\boldsymbol{X}}$, the pseudo-inverse should be intepreted in the least-squares sense.

$\mathbf{C}^{\mathrm{T}}$. The recognition precision, for its part, is just the sum of these two (unnormalized) precisions. As we shall see shortly, it will be convenient to have a name for the normalized precision of the emission:

$$\mathbf{K} := \boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\mathbf{C}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1}. \tag{2.15}$$

Marginalization in jointly Gaussian models. So much for the recognition distribution. What about $\hat{p}(\boldsymbol{y};\boldsymbol{\theta})$? We suspect it will be yet another normal distribution, although we need to make sure. Given this suspicion, and since a normal distribution is completely characterized by its first two cumulants, we use the laws of total expectation,

$$\begin{aligned}\mathbb{E}_{\hat{\boldsymbol{Y}}}\left[\hat{\boldsymbol{Y}}\right] &= \mathbb{E}_{\hat{\boldsymbol{X}}}\left[\mathbb{E}_{\hat{\boldsymbol{Y}}|\hat{\boldsymbol{X}}}\left[\hat{\boldsymbol{Y}}|\hat{\boldsymbol{X}}\right]\right] \\ &= \mathbb{E}_{\hat{\boldsymbol{X}}}\left[\mathbf{C}\hat{\boldsymbol{X}} + \boldsymbol{c}\right] \\ &= \mathbf{C}\boldsymbol{\mu}_{\hat{x}} + \boldsymbol{c} =: \boldsymbol{\mu}_{\hat{y}},\end{aligned} \tag{2.16}$$

and of total covariance,

$$\begin{aligned}\mathrm{Cov}_{\hat{\boldsymbol{Y}}}\left[\hat{\boldsymbol{Y}}\right] &= \mathrm{Cov}_{\hat{\boldsymbol{X}}}\left[\mathbb{E}_{\hat{\boldsymbol{Y}}|\hat{\boldsymbol{X}}}\left[\hat{\boldsymbol{Y}}|\hat{\boldsymbol{X}}\right]\right] + \mathbb{E}_{\hat{\boldsymbol{X}}}\left[\mathrm{Cov}_{\hat{\boldsymbol{Y}}|\hat{\boldsymbol{X}}}\left[\hat{\boldsymbol{Y}}|\hat{\boldsymbol{X}}\right]\right] \\ &= \mathrm{Cov}_{\hat{\boldsymbol{X}}}\left[\mathbf{C}\hat{\boldsymbol{X}} + \boldsymbol{c}\right] + \mathbb{E}_{\hat{\boldsymbol{X}}}\left[\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}\right] \\ &= \mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\hat{y}|\hat{x}} =: \boldsymbol{\Sigma}_{\hat{y}}.\end{aligned} \tag{2.17}$$

The higher cumulants of $\hat{\boldsymbol{Y}}$ are necessarily zero because each term in the law of total cumulance invokes a higher-order (than two) cumulant of either $\hat{\boldsymbol{X}}$ or $\hat{\boldsymbol{Y}}|\hat{\boldsymbol{X}}$, and these are all zero. So the marginal distribution is indeed again normal:

$$\hat{p}(\boldsymbol{y};\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{y};\boldsymbol{\mu}_{\hat{y}},\boldsymbol{\Sigma}_{\hat{y}}\big). \tag{2.18}$$

Between this marginal distribution, Eqs. 2.18, 2.16, and 2.17, and the recognition distribution, Eqs. 2.12, 2.13, and 2.14, we have "reversed the arrow" in Fig. 2.2A, providing the alternative parameterization in the equivalent graphical model of Fig. 2.2B.

The source-emission covariance. There is a third convenient way to characterize the joint distribution, and that is to give the distribution of the vector of $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$ concatenated together. Unsurprisingly, this vector is also normally distributed, and indeed we have almost all the pieces for this already in hand.[5] The mean is given by the concatenation of $\boldsymbol{\mu}_{\hat{x}}$ and $\boldsymbol{\mu}_{\hat{y}}$. The covariance matrix has $\boldsymbol{\Sigma}_{\hat{x}}$ and $\boldsymbol{\Sigma}_{\hat{y}}$ in its upper-left and lower-right blocks. What remains is the upper-right (or its transpose in the lower-left) block, $\boldsymbol{\Sigma}_{\hat{x},\hat{y}}$, i.e. the covariance between $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$.

From Eq. 2.11, we can write $\hat{\boldsymbol{Y}} = \mathbf{C}\hat{\boldsymbol{X}} + \boldsymbol{c} + \hat{\boldsymbol{Z}}$, where $\hat{\boldsymbol{Z}} \sim \mathcal{N}\Big(\mathbf{0}, \boldsymbol{\Sigma}_{\hat{y}|\hat{x}}\Big)$ is independent

[5] Notice that the concatenation of $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$ does *not* have a convenient form for the Gaussian mixture model.

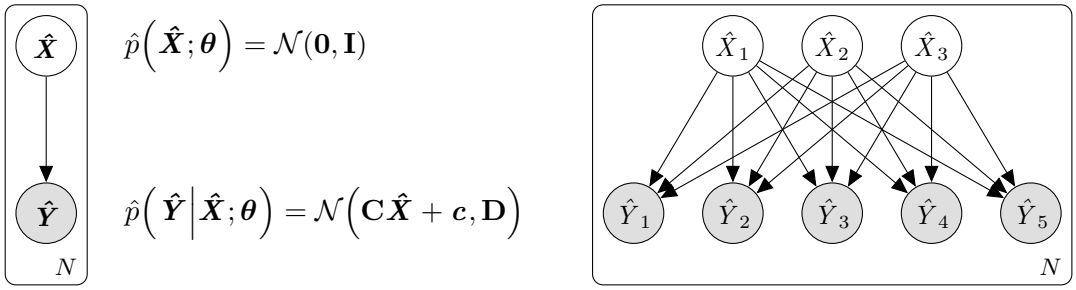


Figure 2.3: **The factor analyzer.** (A) The most abstract version of the model (cf. the GMM in Fig. 2.1). (B) The independence statements made explicit.

of $\hat{\boldsymbol{X}}$. Hence:

$$\begin{aligned}
\mathrm{Cov}\Big[\hat{\boldsymbol{X}},\,\hat{\boldsymbol{Y}}\Big] &= \mathbb{E}\Big[\hat{\boldsymbol{X}}\,\hat{\boldsymbol{Y}}^{\mathrm{T}}\Big] - \mathbb{E}\Big[\hat{\boldsymbol{X}}\Big]\mathbb{E}\Big[\hat{\boldsymbol{Y}}\Big]^{\mathrm{T}} \\
&= \mathbb{E}\Big[\hat{\boldsymbol{X}}(\mathbf{C}\hat{\boldsymbol{X}}+\boldsymbol{c}+\hat{\boldsymbol{Z}})^{\mathrm{T}}\Big] - \mathbb{E}\Big[\hat{\boldsymbol{X}}\Big]\mathbb{E}\Big[\mathbf{C}\hat{\boldsymbol{X}}+\boldsymbol{c}+\hat{\boldsymbol{Z}}\Big]^{\mathrm{T}} \\
&= \mathbb{E}\Big[\hat{\boldsymbol{X}}\,\hat{\boldsymbol{X}}\Big]\mathbf{C}^{\mathrm{T}} + \mathbb{E}\Big[\hat{\boldsymbol{X}}\Big]\boldsymbol{c}^{\mathrm{T}} + \mathbb{E}\Big[\hat{\boldsymbol{X}}\Big]\mathbb{E}\Big[\hat{\boldsymbol{Z}}\Big]^{\mathrm{T}} - \mathbb{E}\Big[\hat{\boldsymbol{X}}\Big]\mathbb{E}\Big[\hat{\boldsymbol{X}}\Big]\mathbf{C}^{\mathrm{T}} - \mathbb{E}\Big[\hat{\boldsymbol{X}}\Big]\boldsymbol{c}^{\mathrm{T}} - \mathbb{E}\Big[\hat{\boldsymbol{X}}\Big]\mathbb{E}\Big[\hat{\boldsymbol{Z}}\Big]^{\mathrm{T}} \\
&= \mathrm{Cov}\Big[\hat{\boldsymbol{X}}\Big]\mathbf{C}^{\mathrm{T}} \\
&= \boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}} =: \boldsymbol{\Sigma}_{\hat{x},\hat{y}}.
\end{aligned} \tag{2.19}$$

This covariance will turn out to be useful later when we take on the learning problem.

Jointly Gaussian random variables and conjugacy. For the models described by Eq. 2.11, it is perhaps unsurprising that the recognition distribution is again normal (Eq. 2.12). After all, the normal distribution is well known to be the conjugate prior for the mean of another normal distribution (it induces a normal posterior distribution over that mean). However, in the family of jointly Gaussian models we are now considering (Eq. 2.11), the mean of the emission distribution is an affine function of, rather than identical with, the source random variable. Evidently, conjugacy holds for this more generic class as well as the classical result for a normally distributed mean, which can be construed as a special case.

include a plot of example data

Factor analysis. We consider now the special case where the underlying normal latent variable $\hat{\boldsymbol{X}}$ is never observed. For example, suppose that $\boldsymbol{Y}$ is the (random) vector of performances on a battery of intelligence tests of various kinds. Each sample $\boldsymbol{y}_n$ corresponds to individual n's performance on all L exams. We *hypothesize* that this performance can be explained by a smaller, $K < L$, number of latent intelligence "factors," $\hat{\boldsymbol{X}}$, that are normally distributed across individuals. But we can't observe them directly; we only observe test scores, which are (ideally) an affine function of these latent factors. Finally, we allow for errors in the model, errors in our measurements, unmodelled enviromental inputs, etc. by allowing our test results to have been corrupted by Gaussian noise.

include the more explicit graphical model....

This sounds like the jointly Gaussian model just discussed, but the fact that we never observe $\hat{\boldsymbol{X}}$ obliges us to restrict the degress of freedom of the model, as discussed at the

outset of this chapter. In particular, we have just seen that the marginal distribution of observed data is normal, Eq. 2.18, and in this factor analysis, this is the only distribution we ever see. Eq. 2.16 shows that, as far as this distribution is concerned, any change in the mean of the source distribution, $\boldsymbol{\mu}_{\hat{x}}$, could equally well be attributed to a change in the emission offset, $\boldsymbol{c}$. So without loss of generality, we can let $\boldsymbol{\mu}_{\hat{x}} = 0$.

A similar consideration applies to the source covariance. From Eq. 2.17, we see that any non-isotropic covariance could simply be absorbed into $\mathbf{C}$. Indeed, this makes the intepretation of the model more intuitive: we are asking for the *independent* factors that together explain test performances. Any covariance across different kinds of intelligence tests is due to their having similar "loading" onto these independent factors.

[[We remove one more degree of freedom. In general, $\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}$ could be any positive definite matrix, but assuming it to be diagonal, $\mathbf{D}$, yields a useful interpretation.....]].

So our complete model is:

$$\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \mathcal{N}(\mathbf{0},\mathbf{I}), \qquad \hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \mathcal{N}\Big(\mathbf{C}\hat{\boldsymbol{X}}+\boldsymbol{c},\mathbf{D}\Big). \tag{2.20}$$

In what follows, we shall assume that one has the correct model, and needs to solve only the inference problem. We shall return to learning in Section 6.4.3.

Inference and marginalization in a factor analyzer. We take the marginal probability first this time. Applying Eqs. 2.16, 2.17, and 2.18 to the special case of Eq. 2.20, we have

$$\hat{p}(\boldsymbol{y};\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{c},\mathbf{C}\mathbf{C}^{\mathrm{T}}+\mathbf{D}\big). \tag{2.21}$$

Inference is just a special case of inference in the jointly Gaussian model introduced at the beginning of this section. That is, it requires only the computation of the recognition cumulants, Eqs. 2.13 and 2.14, under the distributions that define the model, Eq. 2.20:

$$\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\boldsymbol{y};\boldsymbol{\theta}\Big) = \mathcal{N}\Big(\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\mathbf{C}^{\mathrm{T}}\mathbf{D}^{-1}(\boldsymbol{y}-\boldsymbol{c}),\big(\mathbf{C}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{C}+\mathbf{I}\big)^{-1}\Big). \tag{2.22}$$

Anticipating the computational costs of this, and more complicated, inference problems with Gaussian random variables, we note with some dismay the matrix inversions, which cost at least $\mathcal{O}(M^{2.373})$ for $M \times M$ matrices. In factor analysis, as we have noted, the observations $\boldsymbol{y}$ are higher-dimensional than the latent variables $\hat{\boldsymbol{X}}$. It is fortunate, then, that the only matrix inversion carried out in the observation space is of $\mathbf{D}$, which is just $\mathcal{O}(M)$ (invert the entries on the diagonal). But what about models where the "source space" is larger than the "emission space"? What if the emission covariance is not diagonal? And how can we limit the total number of inversions when the inference step has to be applied repeatedly?

Alternative expressions for the recognition cumulants. To change the space in which the recognition covariance is computed, we can apply the Woodbury matrix-inversion lemma, Eq. A.18, to Eq. 2.14:

$$\begin{aligned}\boldsymbol{\Sigma}_{\hat{x}|\hat{y}} &= \Big(\mathbf{C}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1}\mathbf{C}+\boldsymbol{\Sigma}_{\hat{x}}^{-1}\Big)^{-1} \\ &= \boldsymbol{\Sigma}_{\hat{x}}-\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}}(\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}}+\boldsymbol{\Sigma}_{\hat{y}|\hat{x}})^{-1}\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}.\end{aligned} \tag{2.23}$$

This moves the inversion from source ($\hat{\boldsymbol{X}}$) to emission ($\hat{\boldsymbol{Y}}$) space. (Notice, incidentally, that the inverted quantity is precisely the marginal covariance, $\boldsymbol{\Sigma}_{\hat{y}}$.)

The expression for the recognition mean, Eq. 2.13, likewise invokes an inversion in source space ($\boldsymbol{\Sigma}_{\hat{x}}^{-1}$), as well as one in the emission space ($\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1}$). In factor analysis, the first inversion was not required because the source mean, $\boldsymbol{\mu}_{\hat{x}}$, was assumed to be zero. Here we eliminate it by substituting the "Woodburied" recognition covariance, Eq. 2.23, into Eq. 2.13. We also make use of the definition of the emission precision, Eq. 2.15:

$$\begin{aligned}
\boldsymbol{\mu}_{\hat{x}|\hat{y}} &= \boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\Big[\mathbf{C}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1}\Big(\hat{\boldsymbol{Y}} - \boldsymbol{c}\Big) + \boldsymbol{\Sigma}_{\hat{x}}^{-1}\boldsymbol{\mu}_{\hat{x}}\Big] \\
&= \mathbf{K}\Big(\hat{\boldsymbol{Y}} - \boldsymbol{c}\Big) + \boldsymbol{\Sigma}_{\hat{x}|\hat{y}}(\boldsymbol{\Sigma}_{\hat{x}}^{-1})\boldsymbol{\mu}_{\hat{x}} \\
&= \mathbf{K}\Big(\hat{\boldsymbol{Y}} - \boldsymbol{c}\Big) + \boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\Big(\mathbf{C}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1}\mathbf{C} - \mathbf{C}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1}\mathbf{C} + \boldsymbol{\Sigma}_{\hat{x}}^{-1}\Big)\boldsymbol{\mu}_{\hat{x}} \\
&= \mathbf{K}\Big(\hat{\boldsymbol{Y}} - \boldsymbol{c}\Big) - \mathbf{K}\mathbf{C}\boldsymbol{\mu}_{\hat{x}} + \boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\Big(\mathbf{C}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1}\mathbf{C} + \boldsymbol{\Sigma}_{\hat{x}}^{-1}\Big)\boldsymbol{\mu}_{\hat{x}} \\
&= \mathbf{K}\Big(\hat{\boldsymbol{Y}} - \boldsymbol{c}\Big) - \mathbf{K}\mathbf{C}\boldsymbol{\mu}_{\hat{x}} + \boldsymbol{\mu}_{\hat{x}} \\
&= \boldsymbol{\mu}_{\hat{x}} + \mathbf{K}\Big[\hat{\boldsymbol{Y}} - (\mathbf{C}\boldsymbol{\mu}_{\hat{x}} + \boldsymbol{c})\Big].
\end{aligned} \tag{2.24}$$

Intuitively, the recognition mean is the source mean, adjusted by a "correction factor." The correction measures the difference between the emission as observed ($\hat{\boldsymbol{Y}}$) and as predicted by the source mean ($\mathbf{C}\boldsymbol{\mu}_{\hat{x}} + \boldsymbol{c}$), weighted by the precision of the emission ($\mathbf{K}$).

Despite appearances, matrix inversions are still required in Eq. 2.24; they occur in the calculation of $\mathbf{K}$. Consulting its definition, Eq. 2.15, we see one inversion of the emission covariance, and another in the computation of the recognition covariance, albeit in the space of emissions via Eq. 2.23. With a little work, one of these inversions can be eliminated:

$$\begin{aligned}
\mathbf{K} = \boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\mathbf{C}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1} &= \Big[\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}} - \boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}}\Big(\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\hat{y}|\hat{x}}\Big)^{-1}\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}}\Big]\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1} \\
&= \boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}}\Big[\mathbf{I} - \Big(\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\hat{y}|\hat{x}}\Big)^{-1}\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}}\Big]\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1} \\
&= \boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}}(\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\hat{y}|\hat{x}})^{-1}\Big[\Big(\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\hat{y}|\hat{x}}\Big) - \mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}}\Big]\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}^{-1} \\
&= \boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}}(\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\hat{y}|\hat{x}})^{-1}.
\end{aligned} \tag{2.25}$$

Thus in fact only one matrix need be inverted in computing the entire recognition distribution, namely $\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\hat{y}|\hat{x}}$—which, by Eq. 2.17, is just the marginal covariance, $\boldsymbol{\Sigma}_{\hat{y}}$. So we can also write

$$\mathbf{K} = \boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}}^{-1}. \tag{2.26}$$

To make the complete calculation more perspicuous, we collect here the marginal cumulants from Eqs. 2.16 and 2.17:

$$\begin{aligned}
\boldsymbol{\mu}_{\hat{y}} &= \mathbf{C}\boldsymbol{\mu}_{\hat{x}} + \boldsymbol{c}, \\
\boldsymbol{\Sigma}_{\hat{y}} &= \mathbf{C}\boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\hat{y}|\hat{x}},
\end{aligned} \tag{2.27}$$

along with this alternative ("Woodburied") computation of the recognition cumulants:

$$\begin{aligned}
\mathbf{K} &= \boldsymbol{\Sigma}_{\hat{x}}\mathbf{C}^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{y}}^{-1} \\
\boldsymbol{\mu}_{\hat{x}|\hat{y}} &= \boldsymbol{\mu}_{\hat{x}} + \mathbf{K}\Big(\hat{\boldsymbol{Y}} - \boldsymbol{\mu}_{\hat{y}}\Big) \\
\boldsymbol{\Sigma}_{\hat{x}|\hat{y}} &= \boldsymbol{\Sigma}_{\hat{x}} - \mathbf{K}\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}} \\
&= \boldsymbol{\Sigma}_{\hat{x}} - \mathbf{K}\boldsymbol{\Sigma}_{\hat{y}}\mathbf{K}^{\mathrm{T}},
\end{aligned} \tag{2.28}$$

where on the last line we have rewritten the recognition covariance yet again using Eq. 2.26.

2.1.3 Sparse coding

To motivate "sparse" codes, we consider more complicated marginal distributions of data, focusing in particular on the distribution of natural images, for which mixture models and factor analyzers are inadequate models. To see this, we (following Hinton and Ghahramani [10]) cast these two model types as poles of a continuum of models, from sparsest to densest. As we saw above, one possible formulation of mixture models interprets the "source" variable, $\hat{\boldsymbol{X}}$, as a one-hot vector of length K (as opposed to a scalar with support in the set $\{1, \ldots, K\}$). We can even think of each element $\hat{X}_k$ of $\hat{\boldsymbol{X}}$ as a separate "unit" in a neural network. Indeed, the winner-take-all neural networks of the 1980s and '90s [7, Chapter 9] can be thought of as (perhaps approximate) recognition models for generative models that are mixtures. From this perspective, then, the term "mixture" is potentially misleading: the marginal distribution of observations $\hat{\boldsymbol{Y}}$ is indeed a mixture, whence the name, but each individual example $\hat{\boldsymbol{y}}$ is *not* mixed. That is, each $\hat{\boldsymbol{y}}$ is generated by just one element of $\hat{\boldsymbol{x}}$, and the code for generating it is maximally *sparse*.

At the other extreme, in factor analyzers, all of the "hidden units" (elements of $\hat{\boldsymbol{X}}$) typically will have non-zero values, since they are independent and normally distributed. *All* elements of a sample $\hat{\boldsymbol{x}}$ therefore potentially contribute to a single emission $\hat{\boldsymbol{y}}$. This is a *dense* code.

To make matters even more concrete, consider a GMM whose emission covariance is fixed across classes, so that the activity of the source variables determines only the mean of the emission. Then the emissions for the GMM and factor analyzer can be written identically, as a normal distribution about $\mathbf{C}\hat{\boldsymbol{X}}$ (cf. Eq. 2.5 and Eq. 2.20). That is, both models generate data as a "combination" of basis vectors, the columns of $\mathbf{C}$, but differ in how many columns are allowed to contribute to a single observation.

The statistics of natural scenes. Both types of model are problematic for complex distributions, like that of natural images. Gaussian mixture models can indeed approximate any marginal density over $\boldsymbol{Y}$ (Eq. 2.10) to arbitrary accuracy [26]: intuitively, a distribution of any shape can be constructed by placing little blobs of probability mass at various locations. This is essential for the distribution of images, where the presence of (e.g.) lines makes the marginal distribution of (say) triplets of adjacent pixels inexplicable by their mean values or even pairwise correlations alone (Fig. **??**). But each such bit of structure must be "explained" with a separate basis vector! There is no way to explain a letter "X" as a superimposition of two lines; rather, the "X" and the two lines into which it might be decomposed would each require its own basis vector. More generally, every image (or image patch) that could not be expressed as a Gaussian disturbance of some other image would require another column in $\mathbf{C}$. Thus the ability to model structure comes at the price of a very large number of mixture components. It must be emphasized that we believe this price to be too high precisely because we believe the structure in images to be "componential," explicable in terms of combinations of simpler basis elements.

A factor analyzer, in contrast, can avail itself of all basis vectors at once in attempting to explain or generate an observation, and consequently can in theory recognize an image in terms of its constituent parts. By the same token, it needs no more hidden units than there are pixels in the images: $\mathbf{C}$ can be square, i.e. a complete basis. However, the marginal distribution of observations under a factor analyzer is just another normal distribution

(recall Eq. 2.21)—a fact that looked helpful from the perspective of calculating with the model, but is fatal for modeling natural images. This is a consequence of the higher-order structure in images alluded to above.

The existence of such higher-order structure as seen in Fig. **??** is not perhaps obvious *a priori*. After all, the size of the covariance matrix for the model marginal, $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$, scales quadratically in the number of pixels—a lot of representational power to work with! For example, the covariance matrix of even modestly sized, (256×256)-pixel, images has $256^2(256^2-1)/2 \approx 2 \times 10^9$ independent entries. It would seem that we could represent any distribution with this many parameters. But in fact, by taking certain symmetries into account, we reduce the number of covariance entries to a linear function of the number of pixels. The statistics of natural images are, for the most part[6], stationary: the correlation between any two pairs of pixels depends on the distance between them but not their absolute locations. In our example, there are only 256^2 distances—from, say, the upper left pixel to any pixel in the image, including itself. So the covariance cannot encode so much information after all.

Covariance matrices arising from shift-invariant data will in general have this "symmetric Toeplitz" structure. More specifically, when the shift-invariant data are two-dimensional, like images, the covariance matrix will be block-Toeplitz with Toeplitz blocks [5]. Furthermore, if we assume that the covariance drops to zero at some distance within the image, and we ignore edge effects, the covariance matrix can be approximated as block-circulant with circulant blocks, in which case its eigenvectors are the (two-dimensional) discrete Fourier modes, and its eigenvalues constitute the power spectrum [5].[7] In natural images, the power spectrum appears to fall off with frequency f at a rate of very nearly $1/f^2$ [22].

Thus, natural images can be decorrelated by projection onto the Fourier modes, and then "whitened" by normalizing by the power spectrum. Yet it is easily seen Fig. **??** that this process leaves most of the structure intact [22]. Or again, coming from the other direction, restoring the (approximately) $1/f^2$ power spectrum to white noise does not produce a natural image (Fig. **??**). Factor analyzers thus have no means of capturing the structure in natural images that is lies beyond the mean and the power spectrum.

Moderately sparse generative models. We have established, then, that neither factor analyzers nor GMMs are good models for natural images, albeit for roughly opposite reasons. Furthermore, the inadequacy of the multivariate Gaussian as a marginal distribution for natural images is relevant to a wider range of models than the factor analyzer. When the source variables are densely, even though non-normally, distributed, the model marginal $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$ may very well end up approximately normal anyway, for central-limit-like reasons: the marginal distribution is an average of many independent, "random" basis vectors. Averaging can wash out higher-order structure in these vectors. This is (I take it) another way of expressing Hinton's conjecture that the most useful codes will be componential, in order to represent the properties—pose, deformation, etc.—of the objects in the image; but also non-dense, in order to accommodate multiple objects in the same image [10].

Between the "extremes" of the GMM and the factor analyzer lie models in which the

[6] Many natural images, as seen by land-dwelling animals, have sky in the upper portion and ground below, which does introduce statistical regularities that depend on absolute position. Overrepresentation of vertical and (especially) horizontal lines would likewise introduce assymmetries.

[7] Since the eigenvectors of the covariance matrix are the *principal components* of the data (see also Section 6.4.3), this likewise implies that the Fourier modes are the principal components of shift-invariant data.

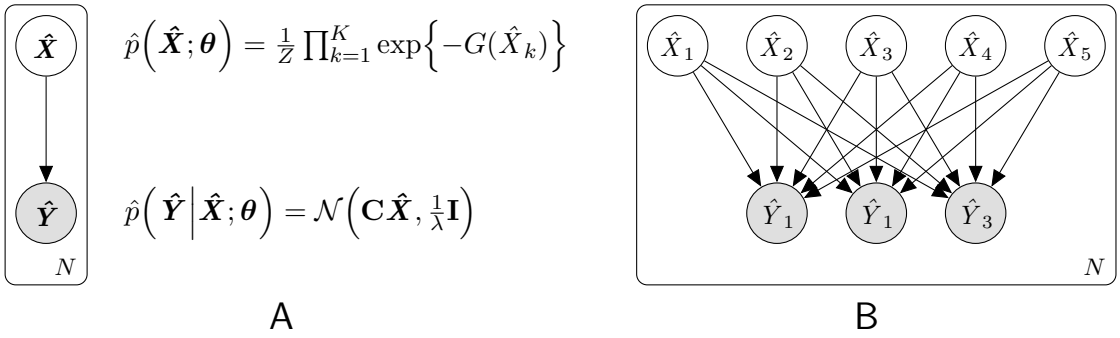


Figure 2.4: **A sparse-coding model.** (A) The most abstract version of the model (cf. the GMM in Fig. 2.1 and the factor analyzer in Fig. 2.3A). The energy function G enforces sparseness; e.g., $G(\hat{X}_k) = \alpha_k\Big|\hat{X}_k\Big|$, or $G(\hat{X}_k) = \log\Big(\beta^2 + \hat{X}_k^2\Big)$ (B) The independence statements and overcompleteness made explicit.

"source" variables are sparsely distributed: some small subset are "on" for any given image. For example, they could be distributed according to a product over Bernoulli distributions with small means. Or again, we could interpret "on" more liberally and assign fat-tailed (leptokurtotic) distributions to the source variables, so that they usually take on values close to zero but can take on extreme probabilities more often than normally distributed variables. Common choices are the Laplace and Cauchy distributions [18, 14]. In this case it is convenient to write the source distribution in the form of a product of Boltzmann distributions,

$$\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \frac{1}{Z(\boldsymbol{\theta})}\prod_{k=1}^{K}\exp\Big\{-E_k(\hat{X}_k,\boldsymbol{\theta})\Big\}, \qquad \hat{p}\Big(\hat{\boldsymbol{Y}}\,\Big|\,\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \mathcal{N}\Big(\mathbf{C}\hat{\boldsymbol{X}}, \frac{1}{\lambda}\mathbf{I}\Big), \tag{2.29}$$

with the energy functions E_k chosen so as to enforce some form of sparseness. For example,

$$E_k(\hat{X}_k) := \begin{cases} \log\Big(\beta_k^2 + \hat{X}_k^2\Big) & \text{Cauchy distribution} \\ \alpha_k\Big|\hat{X}_k\Big| & \text{Laplace distribution} \\ \frac{\alpha_k}{\beta}\log\Big(\cosh(\beta\hat{X}_k)\Big) & \text{hyperbolic-cosine distribution.} \end{cases} \tag{2.30}$$

The last is not a standard distribution, but for large β it approximates the Laplace distribution while retaining differentiability at 0 [15]; see Fig. **??**. It will serve as our default example in the sparse-coding model that follows. The emissions, for their part, are once again normally distributed about a linear function of the source variables, and for simplicity we have given them identical variance, $1/\lambda$. Eq. 2.29 describes the complete sparse-coding model, a graphical model for which is shown in Fig. 2.4A.

Since we are aiming at economy of expression (few descriptors per image), we must concede prodigality in vocabulary (many descriptors) [21]—i.e., an overcomplete set of basis vectors. This is made explicit in Fig. 2.4B, along with the independence statements implied by our choices of source and emission distributions. A factor of 2 overcompleteness is common [14].

...

Inference and marginalization in a sparse-coding model: Laplace's method. There is a reason that factor analyzers and GMMs remain popular despite their limitations: the model recognition and marginal distributions are computable in closed form. This computability is due to the source distributions being conjugate (or pseudo-conjugate) to the emission "likelihoods." When they are not, the marginal and recognition distributions can typically be computed only approximately. When the source variables are Laplace-distributed, e.g., it is not clear that the recognition distribution can be computed (although see [20]).

Perhaps the simplest approach is to make a Taylor approximation.[8] In particular, consider approximating the energy, $E(\hat{\boldsymbol{x}})$, of a distribution near its mode, $\hat{\boldsymbol{x}}^0$, with the first three terms of the Taylor series. It is easy to see that, for $\hat{\boldsymbol{x}}^0$ to be a mode of the distribution, it must satisfy the following conditions:

$$\frac{\partial E}{\partial \hat{\boldsymbol{x}}^{\mathrm{T}}}(\hat{\boldsymbol{x}}^0) = 0, \qquad\qquad \frac{\partial^2 E}{\partial \hat{\boldsymbol{x}} \partial \hat{\boldsymbol{x}}^{\mathrm{T}}}(\hat{\boldsymbol{x}}^0) \succ 0;$$

i.e., at the mode, the energy's gradient must be zero and its Hessian must be positive definite. For a generic energy, the second-order Taylor approximation at the mode is therefore

$$\begin{aligned} E(\hat{\boldsymbol{X}}) &\approx E(\hat{\boldsymbol{x}}^0) + \frac{\partial E}{\partial \hat{\boldsymbol{x}}^{\mathrm{T}}}(\hat{\boldsymbol{x}}^0)\Big(\hat{\boldsymbol{X}} - \hat{\boldsymbol{x}}^0\Big) + \frac{1}{2}\Big(\hat{\boldsymbol{X}} - \hat{\boldsymbol{x}}^0\Big)^{\mathrm{T}} \frac{\partial^2 E}{\partial \hat{\boldsymbol{x}} \partial \hat{\boldsymbol{x}}^{\mathrm{T}}}(\hat{\boldsymbol{x}}^0)\Big(\hat{\boldsymbol{X}} - \hat{\boldsymbol{x}}^0\Big) \\ &= E(\hat{\boldsymbol{x}}^0) + \frac{1}{2}\Big(\hat{\boldsymbol{X}} - \hat{\boldsymbol{x}}^0\Big)^{\mathrm{T}} \frac{\partial^2 E}{\partial \hat{\boldsymbol{x}} \partial \hat{\boldsymbol{x}}^{\mathrm{T}}}(\hat{\boldsymbol{x}}^0)\Big(\hat{\boldsymbol{X}} - \hat{\boldsymbol{x}}^0\Big). \end{aligned} \tag{2.31}$$

The only remaining $\hat{\boldsymbol{X}}$-dependent term should look familiar: it is the energy of a normal distribution with mean at the mode and inverse covariance equal to the Hessian of the energy (evaluated at the mode). The first term, $E(\hat{\boldsymbol{x}}^0)$, on the other hand, is constant in the random variable $\hat{\boldsymbol{X}}$ and can therefore be absorbed into the normalizer. Hence we can approximate the recognition distribution as

$$\hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big) = \frac{1}{Z}\exp\Big\{-E_{\mathrm{recog}}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}, \boldsymbol{\theta})\Big\} \approx \mathcal{N}\left(\hat{\boldsymbol{x}}^0, \left(\frac{\partial^2 E_{\mathrm{recog}}}{\partial \hat{\boldsymbol{x}} \partial \hat{\boldsymbol{x}}^{\mathrm{T}}}(\hat{\boldsymbol{x}}^0, \hat{\boldsymbol{Y}})\right)^{-1}\right). \tag{2.32}$$

Note that this approximation is valid at realizations of $\hat{\boldsymbol{X}}$ even "moderately" close to $\hat{\boldsymbol{x}}^0$, since the neglected higher-order terms need only be less than unity in order for the exponentiation to drive their contribution to the distribution down to much smaller values. On the other hand, it is obviously inappropriate for distributions having multiple modes, or over discrete random variables.

So much for approximating the recognition distribution; we now move on to the model marginal. Using the definition of conditional probability with the approximate recognition

[8] Another, more powerful class of techniques for approximate inference is based on (iterative) *optimization* of the recognition distribution. However, this endows it with the character of learning, and accordingly we shall defer discussing these techniques until Chapter 6.

distribution, Eq. 2.32, and then evaluating at the mode, we see that

$$\begin{aligned}
\hat{p}\Big(\hat{\boldsymbol{X}},\, \hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) &= \hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) \\
&\approx \mathcal{N}\left(\hat{\boldsymbol{x}}^0,\, \left(\frac{\partial^2 E_{\text{recog}}}{\partial\hat{\boldsymbol{x}}\,\partial\hat{\boldsymbol{x}}^{\mathrm{T}}}(\hat{\boldsymbol{x}}^0,\, \hat{\boldsymbol{Y}})\right)^{-1}\right)\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) \\
\implies \hat{p}\Big(\hat{\boldsymbol{x}}^0,\, \hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) &\approx \tau^{-k/2}\left|\frac{\partial^2 E_{\text{recog}}}{\partial\hat{\boldsymbol{x}}\,\partial\hat{\boldsymbol{x}}^{\mathrm{T}}}(\hat{\boldsymbol{x}}^0,\, \hat{\boldsymbol{Y}})\right|^{1/2}\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) \\
\implies \hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) &= \hat{p}\Big(\hat{\boldsymbol{x}}^0,\, \hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)\tau^{k/2}\left|\frac{\partial^2 E_{\text{recog}}}{\partial\hat{\boldsymbol{x}}\,\partial\hat{\boldsymbol{x}}^{\mathrm{T}}}(\hat{\boldsymbol{x}}^0,\, \hat{\boldsymbol{Y}})\right|^{-1/2}.
\end{aligned} \tag{2.33}$$

This is the approximate model marginal under Laplace's method.

Let us specialize to an isotropic Gaussian emission and a factorial source distribution, Eq. 2.29. Start with the Hessian:

$$\begin{aligned}
\frac{\partial^2}{\partial\hat{\boldsymbol{x}}\,\partial\hat{\boldsymbol{x}}^{\mathrm{T}}}E_{\text{recog}}(\hat{\boldsymbol{X}},\boldsymbol{\theta}) &= \frac{\partial^2}{\partial\hat{\boldsymbol{x}}\,\partial\hat{\boldsymbol{x}}^{\mathrm{T}}}E_{\text{joint}}(\hat{\boldsymbol{X}},\boldsymbol{\theta}) \\
&= \frac{\partial^2}{\partial\hat{\boldsymbol{x}}\,\partial\hat{\boldsymbol{x}}^{\mathrm{T}}}\left(\sum_{k=1}^{K}E_k(\hat{X}_k,\boldsymbol{\theta}) + \frac{\lambda}{2}\Big\|\boldsymbol{y}-\mathbf{C}\hat{\boldsymbol{X}}\Big\|^2\right) \\
&= \operatorname{diag}\left(\left[\frac{\partial^2 E_1}{\partial\hat{x}_1{}^2}(\hat{X}_1,\boldsymbol{\theta}),\cdots,\frac{\partial^2 E_K}{\partial\hat{x}_K{}^2}(\hat{X}_K,\boldsymbol{\theta})\right]^{\mathrm{T}}\right) + \lambda\mathbf{C}^{\mathrm{T}}\mathbf{C},
\end{aligned}$$

where $\operatorname{diag}(\boldsymbol{a})$ is a diagonal matrix with the vector $\boldsymbol{a}$ as its diagonal. For concreteness, let us specialize further to Laplace-distributed source variables, as in Eq. 2.30. Unfortunately, the curvature of this energy function is undefined at the origin, and (equally bad) zero everywhere else. Lewicki and colleagues [14, 16] therefore suggest approximating this energy—in the Hessian only—with the hyperbolic-cosine energy (see again Eq. 2.30), which approaches the Laplace energy in the limit of large β. The second derivative of this energy is

$$\frac{\partial^2 E_k}{\partial\hat{x}_k{}^2}(\hat{X}_k,\boldsymbol{\theta}) = \frac{\partial^2}{\partial\hat{x}_k{}^2}\frac{\alpha_k}{\beta}\log\Big(\cosh(\beta\hat{X}_k)\Big) = \alpha_k\frac{\partial}{\partial\hat{x}_k}\tanh(\beta\hat{X}_k) = \alpha_k\beta\operatorname{sech}^2(\beta\hat{X}_k).$$

Hence, from Eq. 2.32, an approximate recognition distribution for the sparse-coding model is

$$\hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) \approx \mathcal{N}\Big(\hat{\boldsymbol{x}}_0,\, \big(\lambda\mathbf{C}^{\mathrm{T}}\mathbf{C} + \operatorname{diag}\big(\boldsymbol{\alpha}\circ\beta\operatorname{sech}^2(\beta\hat{\boldsymbol{x}}_0)\big)\big)^{-1}\Big), \tag{2.34}$$

where $\circ$ is the element-wise product and $\operatorname{sech}(\cdot)$ is supposed to act element-wise. Notice that although this may appear to be independent of the observations $\boldsymbol{y}$, they enter through the mode, which obviously varies as a function of $\boldsymbol{y}$.

To specialize the marginal distribution to the Laplace prior, we use Eq. 2.33, substituting the Laplace energies into the joint, but the hyperbolic-cosine energies into the Hessian (to maintain differentiability):

$$\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) \approx \frac{\tau^{k/2}}{Z_{\text{joint}}}\exp\left\{\boldsymbol{\alpha}^{\mathrm{T}}|\hat{\boldsymbol{x}}_0| - \frac{\lambda}{2}\|\,\boldsymbol{Y}-\mathbf{C}\hat{\boldsymbol{x}}_0\|^2\right\}\big|\lambda\mathbf{C}^{\mathrm{T}}\mathbf{C} + \operatorname{diag}\big(\boldsymbol{\alpha}\circ\beta\operatorname{sech}^2(\beta\hat{\boldsymbol{x}}_0)\big)\big|^{-1/2}. \tag{2.35}$$

...

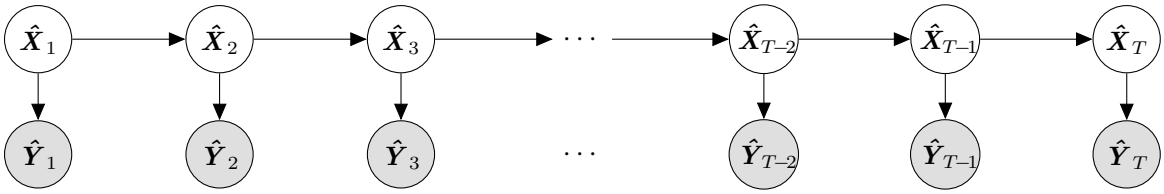


Figure 2.5: **The hidden Markov Model.** Generally, this name is used only if the latent state is discrete, but the same architecture underlies state-space models with real-valued hidden states.

2.2 Dynamical models

Consider the graphical model in Fig. 2.5. Here we finally encounter some interesting statistical-independence structure among the variables. One interpretation is that we have dropped the i.i.d. assumption: the plates that we saw in Figs. 2.1 and 2.3 have been replaced with explicit representation of the source-emission replicates, because now "adjacent" sources are dependent on each other. The joint distribution still factorizes, but in a more complicated way, with individual factors linking together pairs of random variables:

$$\hat{p}\Big(\hat{\boldsymbol{X}}_1, \ldots, \hat{\boldsymbol{X}}_T, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\Big) = \prod_{t=1}^{T} \hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \hat{\boldsymbol{X}}_{t-1}\Big) \hat{p}\Big(\boldsymbol{y}_t \Big| \hat{\boldsymbol{X}}_t\Big). \tag{2.36}$$

(A few simple remarks on the notation: (1) There is no $\hat{\boldsymbol{X}}_0$, but we define $\hat{p}\Big(\hat{\boldsymbol{X}}_1 \Big| \hat{\boldsymbol{X}}_0\Big) := \hat{p}\Big(\hat{\boldsymbol{X}}_1\Big)$, the initial source distribution. This allows for a more compact expression of the joint. (2) The joint has been written as a function of the instantiations of $\boldsymbol{y}_t$ to reflect the assumption that they have been observed. (3) The dependence on parameters has been suppressed to reduce clutter.)

Alternatively, we can interpret Fig. 2.5 as showing structure *internal* to each of N source and emission variables, in which case the entire graph could be wrapped in a plate, and Eq. 2.36 would represent just one sample sequence of many. Within each sequence, we assume that the data were "emitted" by some process with Markov dynamics. We will prefer the second interpretation because it allows for multiple, i.i.d. observation sequences, a scenario typically encountered.

Note the tree structure in Fig. 2.5. It implies that inference—although substantially more complicated than that for the models encountered previously, which was essentially a single application of Bayes's rule—is still only a straightforward application of the sum-product algorithm. Making that connection precise is, however, left as an exercise for the reader. Here we derive, and show the connections between, the classical inference algorithms for the two main models for which this graph structure holds: the "forward-backward" algorithm for the hidden Markov model; and the Kalman filter and Rausch-Tung-Striebel smoother for the state-space model (SSM).

We begin without specializing to either of these models. The general strategy will be to try to derive recursive procedures. More specifically, we will compute the *filter distribution*, $\hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\Big)$, i.e. the distribution over hidden state t given all the observations up to and including point t, in a *forward* pass. Then in a backward pass starting from sample T, we will compute the *smoother distributions*, $\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}, \hat{\boldsymbol{X}}_t | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\Big)$ and $\hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\Big)$: the distribution over hidden states given *all* the observations *sans phrase*. We derive the

smoother for pairs of adjacent bstates as well as single states in anticipation of their usefulness in *learning*. In particular, it is intuitively clear that it would be impossible to learn state-transition probabilities with only distributions over single states.

These are not the only possible forward and backward recursions, and indeed the classical version of the forward-backward algorithm for HMMs computes variations on both of these quantities. We start with the filter and smoother, however, to establish the correspondence with Kalman filtering and RTS smoothing in the case of the SSM. Furthermore, the filter distribution may be useful *per se*, as (e.g.) for temporal data (t indexing the time of arrival) that need to be processed on-line. Finally, for compactness, all marginalizations in the development below are written as sums (as for the HMM) rather than integrals (as for the SSM).

Filtering. We proceed by induction. In particular, we assume that we start with the filter distribution at $t-1$, $\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)$, and show how to update it to $\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_t\Big)$. We will handle the base case at the end.

To derive the recursion, we work with the two graphical-model fragments shown in Fig. 2.6A and Fig. 2.6B. The distributions we have chosen to parameterize these fragments are those associated with the graphical model in Fig. 2.5; that is, they are either factors in the parameterization of the joint distribution in Eq. 2.36, or derivable (we hope) from it. In any case, they provide perfectly legitimate parameterizations for the directed graphical models in Fig. **??**, although why we have chosen precisely these may not yet be clear.

In particular, the filtering process can be thought of as repeated application of two stages, corresponding to operations on these two model fragments: (a) the time update (marginalization in Fig. 2.6A) and (b) the measurement update ("Bayes inversion" in Fig. 2.6B). Thus the time update is:

TIME UPDATE $\quad \hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big), \rightarrow \hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)$

$$\textsc{Marginalize}\begin{pmatrix}\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)\\ \hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{X}}_{t-1}\Big)\end{pmatrix}$$

$$\begin{aligned}\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big) &= \sum_{\hat{\boldsymbol{x}}_{t-1}} \hat{p}\big(\hat{\boldsymbol{x}}_{t-1}\big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\big)\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{x}}_{t-1},\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)\\ &= \sum_{\hat{\boldsymbol{x}}_{t-1}} \hat{p}\big(\hat{\boldsymbol{x}}_{t-1}\big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\big)\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{x}}_{t-1}\Big)\end{aligned} \tag{2.37}$$

The second equality follows from properties of the graph in Fig. 2.5: given the previous state, the current state is independent of all preceding observations. Thus we have transformed a (filter) distribution over the previous state, $\hat{\boldsymbol{X}}_{t-1}$, into a distribution over the current state, $\hat{\boldsymbol{X}}_t$ (in both cases conditioned on observations up $t-1$)—hence the "time update."

In the *measurement* update, we will add another measurement to the filter distribution; that is, we will transform a distribution over $\hat{\boldsymbol{X}}_t$ by adding the current observation, $\boldsymbol{y}_t$, to the set of conditioning variables. In particular, using Bayes's rule on the graph in Fig. 2.6B to compute the recognition distribution over $\hat{\boldsymbol{X}}_t$:

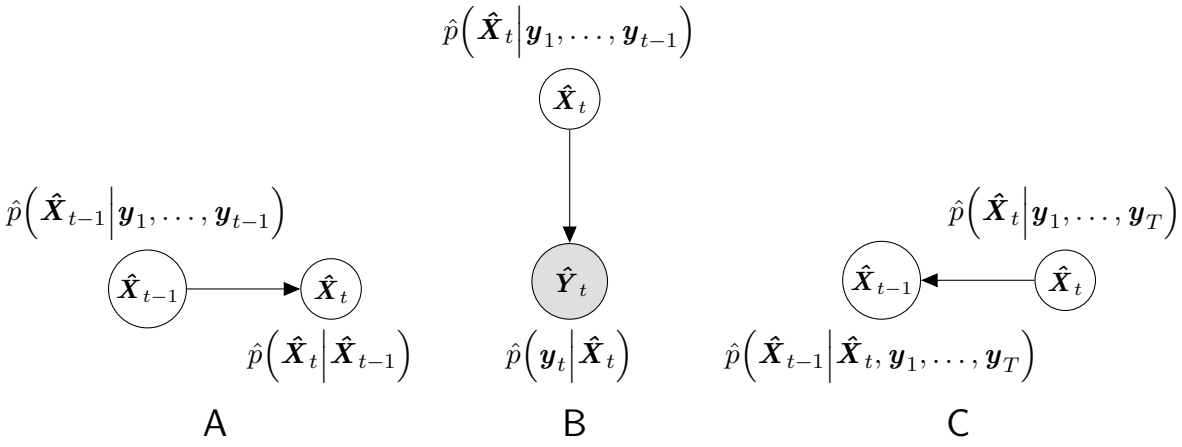


Figure 2.6: **Three graphical model fragments for HMM inference problems.** The various inference problems for HMM-like models can be expressed in terms of recursive application to these three fragments two operations: marginalizing out the source variable (*marginalization*) and computing the recognition distribution (*inversion*). (A) Marginalization: the time update. Inversion: future conditioning. (B) Marginalization: the recursion for the data marginal. Inversion: the measurement update. (C) Marginalization: the backward step. Inversion: the recursion for the recognition distribution over sequences. The distributions have been written in terms of a particular instantiation of the observations (i.e., lowercase letters) to emphasize their use in *inference*.

MEASUREMENT UPDATE $\quad \hat{p}\left(\hat{\boldsymbol{X}}_t \middle| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\right) \to \hat{p}\left(\hat{\boldsymbol{X}}_t \middle| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\right)$

$$\textsc{Bayes-Invert} \begin{pmatrix} \hat{p}\left(\hat{\boldsymbol{X}}_t \middle| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\right) \\ \hat{p}\left(\boldsymbol{y}_t \middle| \hat{\boldsymbol{X}}_t\right) \end{pmatrix}$$

$$\begin{aligned} \hat{p}\left(\hat{\boldsymbol{X}}_t \middle| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\right) &= \frac{\hat{p}\left(\hat{\boldsymbol{X}}_t \middle| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\right)\hat{p}\left(\boldsymbol{y}_t \middle| \hat{\boldsymbol{X}}_t, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\right)}{\sum_{\hat{\boldsymbol{x}}_t} \hat{p}(\hat{\boldsymbol{x}}_t | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})\hat{p}(\boldsymbol{y}_t | \hat{\boldsymbol{x}}_t, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})} \\ &= \frac{\hat{p}\left(\hat{\boldsymbol{X}}_t \middle| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\right)\hat{p}\left(\boldsymbol{y}_t \middle| \hat{\boldsymbol{X}}_t\right)}{\sum_{\hat{\boldsymbol{x}}_t} \hat{p}(\hat{\boldsymbol{x}}_t | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})\hat{p}(\boldsymbol{y}_t | \hat{\boldsymbol{x}}_t)}. \end{aligned} \tag{2.38}$$

Here the second equality again follows from independence statements asserted by the graph in Fig. 2.5: given the current state, the current observation is independent of all past observations.

Thus Eqs. 2.37 and 2.38 together transform the filter distribution at step $t-1$ into the filter distribution at step t. The question is how difficult the operations in these equations are to carry out. We precise them to the case of the HMM and SSM below. All that remains is the base case, then, and it is indeed obvious that at step $t = 1$, the measurement update is simply initialized at $\hat{p}\left(\hat{\boldsymbol{X}}_1\right)$, the source distribution over the initial states.

Smoothing. Again we break the recursion into two stages, corresponding to Fig. 2.6B and Fig. 2.6C, although here we reverse the order of "Bayes inversion" and marginalization. This time we assume that we start with a smoother distribution at step t and show how to get the smoother distribution at step $t-1$, i.e. a backwards recursion.

In the first stage, we return to the graphical model fragment used in the the time update, Fig. 2.6A, only this time we compute a recognition distribution rather than merely marginalizing. Of course, this "inversion" requires computing the marginal along the way, so we may be able to reuse here some of the computations from the time update. We revisit this point below. *Faute de mieux*, we call this stage "future conditioning":

FUTURE CONDITIONING

$$\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big) \to \hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)$$

$$\textsc{Bayes-Invert}\begin{pmatrix}\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)\\ \hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{X}}_{t-1}\Big)\end{pmatrix}$$

$$\begin{aligned}\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big) &= \hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)\\ &= \frac{\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{X}}_{t-1},\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)}{\sum_{\hat{\boldsymbol{x}}_{t-1}}\hat{p}(\hat{\boldsymbol{x}}_{t-1}|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1})\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{x}}_{t-1},\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)}\\ &= \frac{\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{X}}_{t-1}\Big)}{\sum_{\hat{\boldsymbol{x}}_{t-1}}\hat{p}(\hat{\boldsymbol{x}}_{t-1}|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1})\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{x}}_{t-1}\Big)}.\end{aligned} \tag{2.39}$$

The final equality follows, as in the time update, from the fact that $\hat{\boldsymbol{X}}_t$ is independent of all past observations, conditioned on $\hat{\boldsymbol{X}}_{t-1}$. The first equality follows from the mirror-image independence statement: conditioned on the next state, the current state is independent of all *future* observations.

The second and final stage, which we call the "backward step," marginalizes out $\hat{\boldsymbol{X}}_t$ from the graph fragment in Fig. 2.6C:

BACKWARD STEP

$$\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big) \to \hat{p}\Big(\hat{\boldsymbol{X}}_{t-1},\hat{\boldsymbol{X}}_t|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)$$
$$\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big) \to \hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)$$

$$\textsc{Marginalize}\begin{pmatrix}\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)\\ \hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)\end{pmatrix}$$

$$\begin{aligned}\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big) &= \sum_{\hat{\boldsymbol{x}}_t}\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1},\hat{\boldsymbol{x}}_t|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)\\ &= \sum_{\hat{\boldsymbol{x}}_t}\hat{p}(\hat{\boldsymbol{x}}_t|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T)\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\hat{\boldsymbol{x}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big),\end{aligned} \tag{2.40}$$

yielding the smoother distributions at the previous step $(t-1)$. Notice that the distribution over pairs of states is simply the joint distribution in the penultimate line prior to marginalization.

One interesting fact about the smoother is that it does not invoke $\hat{p}\Big(\boldsymbol{y}_t\Big|\hat{\boldsymbol{X}}_t\Big)$; that is, it does not need to (re-)access the data! Instead, all relevant information about the observations is contained in the filter distribution.

So much for the recursion; what about the initialization? The answer is trivial: at step T, the smoother distribution is identical to the filter distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}_T\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)$.

Marginalizing out the states. The joint distribution $\hat{p}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T)$ can also be computed with a recursion. By the chain rule of probability,

$$\hat{p}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_t) = \prod_{t=1}^{T}\hat{p}(\boldsymbol{y}_t|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}). \tag{2.41}$$

And in fact, we are already required to calculate the factors on the right-hand side for the measurement updates, Eq. 2.38: We saw that each measurement update is an "inversion" of Fig. 2.6B, which requires along the way computing the marginal distribution over $\boldsymbol{y}_t$ in this graph fragment. It turns out that these marginals are the factors in Eq. 2.41:

$$\sum_{\hat{\boldsymbol{x}}_t}\hat{p}(\hat{\boldsymbol{x}}_t|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1})\hat{p}(\boldsymbol{y}_t|\hat{\boldsymbol{x}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}) = \hat{p}(\boldsymbol{y}_t|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}).$$

Inference over sequences. This observation suggests an interesting symmetry. Marginalizing and Bayes inverting in all three of the graphical-model fragments in Fig. 2.6 yields six total operations, five of which we have now made use of: The time update and future conditioning correspond, respectively, to marginalization and inversion in Fig. 2.6A. The recursion for the data marginal and the measurement update correspond respectively, as we have just seen, to marginalization and inversion in Fig. 2.6B. On the remaining graph fragment, Fig. 2.6C, the backward step corresponds to marginalization. That leaves inversion

on this fragment—what does it compute?

$$\frac{\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)}{\sum_{\hat{\boldsymbol{x}}_t}\hat{p}(\hat{\boldsymbol{x}}_t\big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T)\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\hat{\boldsymbol{x}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)} = \hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{X}}_{t-1},\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)$$
$$= \hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{X}}_{t-1},\boldsymbol{y}_t,\ldots,\boldsymbol{y}_T\Big).$$

Now consider the following recursion for computing the recognition distribution over *an entire sequence*, starting again from the chain rule of probability and then applying conditional-independence statements asserted by the graph:

$$\hat{p}\Big(\hat{\boldsymbol{X}}_1,\ldots,\hat{\boldsymbol{X}}_T\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big) = \prod_{t=1}^{T}\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{X}}_1,\ldots,\hat{\boldsymbol{X}}_{t-1},\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)$$
$$= \prod_{t=1}^{T}\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{X}}_{t-1},\boldsymbol{y}_t,\ldots,\boldsymbol{y}_T\Big).$$

So we see that each of the six combinations of marginalization and Bayes inversion on each of the three unique graphical model fragements of Fig. 2.6 is a useful computation in its own right.

2.2.1 The hidden Markov Model and the α-γ algorithm

We have not as yet specified the conditional distributions parameterizing the graph in Fig. 2.5 or Eq. 2.36. Let us specify that the states are discrete and therefore categorically distributed:

$$\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{X}}_{t-1}\Big) = \begin{cases}\text{Categ}(\boldsymbol{\pi}) & t=1\\ \text{Categ}\Big(\mathbf{A}\hat{\boldsymbol{X}}_{t-1}\Big) & t>1\end{cases}$$

where the vector $\boldsymbol{\pi}$ and matrix $\mathbf{A}$ are parameters. Note that we have assumed a *homogeneous* Markov chain, i.e., $\mathbf{A}$ is independent of t.

The most important consequence of this choice for the states of the "backbone" is that the filter and smoother distributions are both again categorical distributions, no matter what the form of the emission densities. This is a result of the "pseudo-conjugacy" of the categorical distribution (see above).

Filtering and smoothing. Now for concreteness, let us consider in turn each of the four half-updates, referring to the corresponding graphical model, as they apply to the HMM. Since the filter distribution is categorical, the joint distribution represented by Fig. 2.6A is a(nother) mixture model. Marginalizing out the class identities in this model—the *time update*—therefore amounts to an application of Eq. 2.10. Similarly, the joint distribution of the model in Fig. 2.6B is another mixture model. The *measurement update* amounts to applying Bayes' theorem to this mixture model—Eq. 2.6.

The smoother follows the same template. The joint distributions in both Fig. 2.6A (as we have just seen) and Fig. 2.6C are mixture models, and the "future conditioning" and "backward step" correspond, respectively, to "Bayes inversion" (Eq. 2.6) and marginalization (Eq. 2.10) in these two models.

The α-γ algorithm. What if we are not actually interested in the filter distribution, that is, if all we want at the end of the day are the smoother distributions, $\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)$ and $\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1},\hat{\boldsymbol{X}}_t|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)$ (at all steps t)? Then it can be wasteful and unnecessary to keep renormalizing in the measurement update, Eq. 2.38. We shall see below that it is *not* wasteful for Gaussian random variables, for which the recognition (cumulants) can be calculated without any explicit calculation of the normalizer. But for discrete random variables, we are nearly doubling the number of operations performed. On the other hand, normalization has agreeable numerical consequences, preventing the numerical underflow that would results from repeated multiplication of numbers less than one. Therefore, we present the following mostly to connect with the historical literature.

Consider, then, calculating only the numerator in Eq. 2.38:

$$\begin{aligned}\hat{p}\Big(\hat{\boldsymbol{X}}_t,\boldsymbol{y}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big) &= \hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)\hat{p}\Big(\boldsymbol{y}_t\Big|\hat{\boldsymbol{X}}_t\Big)\\ &= \sum_{\hat{\boldsymbol{x}}_{t-1}}\hat{p}(\hat{\boldsymbol{x}}_{t-1}|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1})\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{x}}_{t-1}\Big)\hat{p}\Big(\boldsymbol{y}_t\Big|\hat{\boldsymbol{X}}_t\Big)\\ \implies \hat{p}\Big(\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_t\Big) &= \sum_{\hat{\boldsymbol{x}}_{t-1}}\hat{p}(\hat{\boldsymbol{x}}_{t-1},\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1})\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{x}}_{t-1}\Big)\hat{p}\Big(\boldsymbol{y}_t\Big|\hat{\boldsymbol{X}}_t\Big)\end{aligned}$$

where on the second line we have substituted in the time update, Eq. 2.37, and on the third multiplied through by $\hat{p}\big(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\big)$. This can be written more compactly still as

α-Recursion

$$\implies \alpha_t(\hat{\boldsymbol{X}}_t) = \sum_{\hat{\boldsymbol{x}}_{t-1}}\alpha_{t-1}(\hat{\boldsymbol{x}}_{t-1})\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{x}}_{t-1}\Big)\hat{p}\Big(\boldsymbol{y}_t\Big|\hat{\boldsymbol{X}}_t\Big) \tag{2.42}$$

with the aid of the definition

$$\alpha_t(\hat{\boldsymbol{X}}_t) := \hat{p}\Big(\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_t\Big). \tag{2.43}$$

This is the well known α forward recursion.

It remains to show that the smoother algorithm can be adjusted to work with $\alpha_t(\hat{\boldsymbol{X}}_t)$ rather than the filter distribution. This is trivial: simply multiplying the numerator and denominator in the final line of the "future conditioning" stage, Eq. 2.39, by $\hat{p}\big(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\big)$ converts the filter densities into α's. That is to say, the smoother recursion does not need to be altered; it works the same if the filter densities are replaced by α's. To make this explicit and invoke the classical terminology, we assign the Greek letter γ to the smoother distribution,

$$\gamma_t(\hat{\boldsymbol{X}}_t) := \hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big) \tag{2.44}$$

and combine the "future conditioning" and "backward step" into a single recursion. We break it into two pieces to show where the smoother distribution over pairs of states is calculated:

γ-Recursion

$$\begin{aligned}
\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}, \hat{\boldsymbol{X}}_t | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\Big) &= \gamma_t(\hat{\boldsymbol{X}}_t) \frac{\alpha_{t-1}(\hat{\boldsymbol{X}}_{t-1}) \hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \hat{\boldsymbol{X}}_{t-1}\Big)}{\sum_{\hat{\boldsymbol{x}}_{t-1}} \alpha_{t-1}(\hat{\boldsymbol{x}}_{t-1}) \hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \hat{\boldsymbol{x}}_{t-1}\Big)} \\
&= \gamma_t(\hat{\boldsymbol{X}}_t) \frac{\alpha_{t-1}(\hat{\boldsymbol{X}}_{t-1}) \hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \hat{\boldsymbol{X}}_{t-1}\Big) \hat{p}\Big(\boldsymbol{y}_t \Big| \hat{\boldsymbol{X}}_t\Big)}{\alpha_t(\hat{\boldsymbol{X}}_t)} \qquad (2.45) \\
\gamma_{t-1}(\hat{\boldsymbol{X}}_{t-1}) &= \sum_{\hat{\boldsymbol{x}}_t} \hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}, \hat{\boldsymbol{x}}_t | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\Big)
\end{aligned}$$

On the second line we have simply substituted in the formula for the alpha recursion, Eq. 2.42. This avoids recalculating the normalizer, as noted above—but at the price of reintroducing a direct dependency on the observations through $\hat{p}\Big(\boldsymbol{y}_t \Big| \hat{\boldsymbol{X}}_t\Big)$.

Computational complexity of HMM filtering and smoothing.

2.2.2 State-space models, the Kalman filter, and the RTS smoother

If the graphical model of the previous section is parameterized with a jointly Gaussian distribution over *all* the variables, a different, but equally tractable, and equally popular, model results: the state-space model. More specifically, the model can be interpreted as a discrete-time, linear, time-invariant (LTI) system, with additive Gaussian noise on the state transitions and observations:

$$\hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \hat{\boldsymbol{X}}_{t-1}\Big) = \mathcal{N}\Big(\mathbf{A}\hat{\boldsymbol{X}}_{t-1} + \mathbf{B}\boldsymbol{u}_{t-1} + \boldsymbol{a}, \boldsymbol{\Sigma}_{\hat{x}}\Big), \quad \hat{p}\Big(\hat{\boldsymbol{Y}}_t \Big| \hat{\boldsymbol{X}}_t\Big) = \mathcal{N}\Big(\mathbf{C}\hat{\boldsymbol{X}}_t + \boldsymbol{c}, \boldsymbol{\Sigma}_{\hat{y}|\hat{x}}\Big).$$

To connect with the historical literature, we have allowed the time evolution of the latent state to depend on some inputs or controls $\boldsymbol{u}$. But notice that the controls are not random variables, since they are assumed to be observed (in theory, we issue them). We could therefore treat all of $\mathbf{B}\boldsymbol{u}_{t-1} + \boldsymbol{a}$ as a single vector of (time-varying) "parameters." In this derivation, however, we will be explicit and maintain separate identities for these quantities throughout.

As in the hidden Markov model, inference in the state-space model requires a forward sweep, followed by a backward sweep, through ordered samples (space or time). Since we have presented the α-γ version of the forward-backward algorithm for the HMM—i.e., forward filtering followed by backward smoothing—the SSM algorithms are identical at this level of description: Eqs. 2.37, 2.38, 2.39, and 2.40.

Differences arise only when we make precise how these steps are to be implemented. That is, in the case of the SSM, applying these four equations amounts to a series of marginalizations and "Bayes inversions" with *jointly Gaussian random variables*. Fortunately, as we have seen in Section 2.1.2, marginalization and Bayes inversion for jointly Gaussian random variables yield elegant closed-form expressions. Moreover, since the set of jointly Gaussian random variables is closed under these operations, *all random variables of interest remain Gaussian throughout*. Thus keeping track of filter and smoother distributions amounts (merely) to keeping track of mean vectors and covariance matrices; and inference amounts to recursive application of certain matrix-vector operations: for marginal

cumulants, Eq. 2.27; for recognition cumulants, Eq. 2.28; and for the cross-covariance matrix, Eq. 2.19. We discuss their computational complexity below. These algorithms go by the famous names of the *Kalman filter* and the *Rauch-Tung-Striebel smoother*.[9]

Filtering. Recall again that the goal is to derive the "filtering" equation, $\hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\Big)$. Here we translate the time update, Eq. 2.37, and the measurement update, Eq. 2.38, into operations on Gaussian random variables. We have lately noted that this will required keeping track only of recognition mean vectors and covariance matrices, so to facilitate that process we begin by assigning shorthand symbols to the filter cumulants,

$$\bar{\hat{\boldsymbol{x}}}_{t|t} := \mathbb{E}\Big[\hat{\boldsymbol{X}}_t | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\Big], \qquad \boldsymbol{\Upsilon}_{t|t} := \mathrm{Cov}\Big[\hat{\boldsymbol{X}}_t | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\Big]; \tag{2.46}$$

and likewise to the recognition mean and covariance given the observations only up to the previous step:

$$\bar{\hat{\boldsymbol{x}}}_{t|t-1} := \mathbb{E}\Big[\hat{\boldsymbol{X}}_t | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\Big], \qquad \boldsymbol{\Upsilon}_{t|t-1} := \mathrm{Cov}\Big[\hat{\boldsymbol{X}}_t | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\Big]. \tag{2.47}$$

We hope that these are the only parameters we ever need to keep track of. To show that they are, we proceed by induction.

We begin with the induction step, which amounts to a measurement update followed by a time update. Assume that the one-step prediction distribution at step $t-1$ is a normal distribution over $\hat{\boldsymbol{X}}_t$, $\hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\Big) = \mathcal{N}\Big(\bar{\hat{\boldsymbol{x}}}_{t|t-1}, \boldsymbol{\Upsilon}_{t|t-1}\Big)$. Then, since the emission $\hat{p}\Big(\hat{\boldsymbol{Y}}_t \Big| \hat{\boldsymbol{X}}_t\Big)$ is likewise a normal distribution, whose mean is an affine function of $\hat{\boldsymbol{X}}_t$, we can simply apply our equations for the recognition cumulants, Eq. 2.28:

Measurement Update

$$\begin{aligned}
\hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\Big) &= \mathcal{N}\Big(\bar{\hat{\boldsymbol{x}}}_{t|t}, \boldsymbol{\Upsilon}_{t|t}\Big), \\
\mathbf{K}_t &= \boldsymbol{\Upsilon}_{t|t-1}\mathbf{C}^{\mathrm{T}}\Big[\mathbf{C}\boldsymbol{\Upsilon}_{t|t-1}\mathbf{C}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\hat{y}|\hat{x}}\Big]^{-1} \\
\bar{\hat{\boldsymbol{x}}}_{t|t} &= \bar{\hat{\boldsymbol{x}}}_{t|t-1} + \mathbf{K}_t\Big[\boldsymbol{y}_t - \Big(\mathbf{C}\bar{\hat{\boldsymbol{x}}}_{t|t-1} + \boldsymbol{c}\Big)\Big] \\
\boldsymbol{\Upsilon}_{t|t} &= \boldsymbol{\Upsilon}_{t|t-1} - \mathbf{K}_t\mathbf{C}\boldsymbol{\Upsilon}_{t|t-1}.
\end{aligned} \tag{2.48}$$

Let us interpret the cumulants. The update to our existing estimate of the current hidden state, $\bar{\hat{\boldsymbol{x}}}_{t|t-1}$, is a product of a precision, $\mathbf{K}_t$, and an accuracy, $\boldsymbol{y}_t - \Big(\mathbf{C}\bar{\hat{\boldsymbol{x}}}_{t|t-1} + \boldsymbol{c}\Big)$. In particular, the more accurately $\bar{\hat{\boldsymbol{x}}}_{t|t-1}$ predicts the current observation ($\boldsymbol{y}_t$), the less it needs to be updated. On the other hand, we need not be troubled by an inaccurate prediction if the observation is not particularly informative, to wit, when the emission is imprecise relative to the original distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\Big)$. This relative precision is measured by the Kalman gain, $\mathbf{K}_t$, as we saw in Eq. 2.15.

We expect the covariance, for its part, to shrink when we add new information. The more precisely the emission reflects the hidden state ($\mathbf{K}_t$), the larger the fraction of the existing covariance ($\boldsymbol{\Upsilon}_{t|t-1}$) we expect to remove.

[9]The term "Kalman smoother" is sometimes encountered in the literature, but is technically a misattribution.

We turn to the time update, which moves our prediction from $\hat{\boldsymbol{X}}_{t-1}$ to $\hat{\boldsymbol{X}}_t$, without considering any new evidence. It does so by considering the state transition, in particular multiplying the filter distribution $\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big)$ by $\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{X}}_{t-1}\Big)$ and marginalizing out $\hat{\boldsymbol{X}}_{t-1}$ (Eq. 2.37). The state transitions are again normally distributed about a mean that is an affine function of $\hat{\boldsymbol{X}}_{t-1}$, which as we have just seen is itself normally distributed under the filter distribution. Therefore we can compute the marginal distribution over $\hat{\boldsymbol{X}}_t$ simply by applying Eq. 2.27:

Time Update

$$\begin{aligned}\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big) &= \mathcal{N}\Big(\bar{\hat{\boldsymbol{x}}}_{t|t-1},\boldsymbol{\Upsilon}_{t|t-1}\Big),\\ \bar{\hat{\boldsymbol{x}}}_{t|t-1} &= \mathbf{A}\bar{\hat{\boldsymbol{x}}}_{t-1|t-1}+\mathbf{B}\boldsymbol{u}_{t-1}+\boldsymbol{a}\\ \boldsymbol{\Upsilon}_{t|t-1} &= \mathbf{A}\boldsymbol{\Upsilon}_{t-1|t-1}\mathbf{A}^{\mathrm{T}}+\boldsymbol{\Sigma}_{\hat{x}}.\end{aligned} \tag{2.49}$$

Thus, our best estimate of the next state ($\bar{\hat{\boldsymbol{x}}}_{t|t-1}$) is simply our estimate of the current state ($\bar{\hat{\boldsymbol{x}}}_{t-1|t-1}$) passed through the state-transition function. The uncertainty (covariance) compounds: it is our uncertainty about the previous state, as transformed by the state-transition matrix, plus the state-transition uncertainty.

We cannot quite say, however, that the uncertainty grows, even though we are not incorporating any new observations and the state transition is noisy: If the underlying dynamical system is strictly stable, i.e. all its eigenvalues are within the unit circle, then the covariance $\mathbf{A}\boldsymbol{\Upsilon}_{t-1|t-1}\mathbf{A}^{\mathrm{T}}$ will be smaller than $\boldsymbol{\Upsilon}_{t-1|t-1}$; and it could in theory be sufficiently smaller even to account for the additional transition noise ($\boldsymbol{\Sigma}_{\hat{x}}$). Such systems, however, are heading rapidly toward an equilibrium point, and are (perhaps) less common targets for tracking than marginally stable systems, with eigenvalues on the unit circle. For such systems, the uncertainty increases with the time update.

To complete the induction, we need to show that the one-step prediction distribution at step 1 is a normal distribution over $\hat{\boldsymbol{X}}_1$. And indeed, setting this distribution to be the source distribution over $\hat{\boldsymbol{X}}_1$, $\hat{p}\Big(\hat{\boldsymbol{X}}_1\Big)$, achieves that goal. For compactness, then, we define

$$\bar{\hat{\boldsymbol{x}}}_{1|0} := \boldsymbol{\mu}_1, \qquad\qquad \boldsymbol{\Upsilon}_{1|0} := \boldsymbol{\Sigma}_1, \tag{2.50}$$

so that Eq. 2.48 gives the measurement update for *all* time steps. Now the time update, Eq. 2.49, and the measurement update, Eq. 2.48, completely define the filter.

Smoothing. Recall the goal: to get, for all t, $\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)$, where we are conditioning on *all* the observations. Once again we shall show that this distribution is normal, and thus we need only keep track of its first two cumulants:

$$\bar{\hat{\boldsymbol{x}}}_{t|T} := \mathbb{E}\Big[\hat{\boldsymbol{X}}_t|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big], \qquad\qquad \boldsymbol{\Upsilon}_{t|T} := \mathrm{Cov}\Big[\hat{\boldsymbol{X}}_t|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big]. \tag{2.51}$$

In deriving our algorithm, it will also be helpful to keep track of the "future-conditioned smoother," $\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big)$, so we give names to its cumulants, as well:

$$\bar{\hat{\boldsymbol{X}}}_{t-1|t,T} := \mathbb{E}\Big[\hat{\boldsymbol{X}}_{t-1}|\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big] \qquad = \mathbb{E}\Big[\hat{\boldsymbol{X}}_{t-1}|\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big], \tag{2.52}$$

$$\boldsymbol{\Upsilon}_{t-1|t,T} := \mathrm{Cov}\Big[\hat{\boldsymbol{X}}_{t-1}|\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big] \qquad = \mathrm{Cov}\Big[\hat{\boldsymbol{X}}_{t-1}|\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big]. \tag{2.53}$$

Notice that the "future-conditioned" mean is a random variable, because it depends on the random variable $\hat{\boldsymbol{X}}_{t-1}$. (The future-conditioned covariance, although it also seems to depend on a random variable, will turn out to be non-random.) This raises a minor question of how to represent and store this mean, an issue which did not arise for the HMM. We shall revisit this after the derivation of the smoother.

Since the Kalman filter leaves us with $\hat{p}\Big(\hat{\boldsymbol{X}}_T \Big| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\Big)$ at the last step, we start there and recurse backwards. This filter distribution is normal, which takes care of the base case of the induction. The induction step, then, assumes that at step t we have $\hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\Big)$, and concludes by showing that we can get $\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1} \Big| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\Big)$ from it.

Beginning with "future conditioning," Eq. 2.39, we "Bayes invert" the filter distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1} \Big| \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\Big)$, and the state-transition distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \hat{\boldsymbol{X}}_{t-1}\Big)$. Once again, $\hat{\boldsymbol{X}}_{t-1}$ is normally distributed under the filter distribution (as we showed in the derivation of the Kalman filter), and $\hat{p}\Big(\hat{\boldsymbol{X}}_t \Big| \hat{\boldsymbol{X}}_{t-1}\Big)$ is normally distributed about an affine function of $\hat{\boldsymbol{X}}_{t-1}$. So we apply Eq. 2.28 and find:

Future Conditioning

$$\begin{aligned}
\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1} \Big| \hat{\boldsymbol{X}}_t, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\Big) &= \mathcal{N}\Big(\bar{\hat{\boldsymbol{X}}}_{t-1|t,T}, \boldsymbol{\Upsilon}_{t-1|t,T}\Big), \\
\mathbf{J}_{t-1} &= \boldsymbol{\Upsilon}_{t-1|t-1}\mathbf{A}^{\mathrm{T}}(\mathbf{A}\boldsymbol{\Upsilon}_{t-1|t-1}\mathbf{A}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\hat{x}})^{-1} \\
&= \boldsymbol{\Upsilon}_{t-1|t-1}\mathbf{A}^{\mathrm{T}}\boldsymbol{\Upsilon}_{t|t-1}^{-1} \\
\bar{\hat{\boldsymbol{X}}}_{t-1|t,T} &= \bar{\hat{\boldsymbol{x}}}_{t-1|t-1} + \mathbf{J}_{t-1}\Big[\hat{\boldsymbol{X}}_t - \Big(\mathbf{A}\bar{\hat{\boldsymbol{x}}}_{t-1|t-1} + \mathbf{B}\boldsymbol{u}_{t-1} + \boldsymbol{a}\Big)\Big] \\
&= \bar{\hat{\boldsymbol{x}}}_{t-1|t-1} + \mathbf{J}_{t-1}(\hat{\boldsymbol{X}}_t - \bar{\hat{\boldsymbol{x}}}_{t|t-1}) \\
\boldsymbol{\Upsilon}_{t-1|t,T} &= \boldsymbol{\Upsilon}_{t-1|t-1} - \mathbf{J}_{t-1}\mathbf{A}\boldsymbol{\Upsilon}_{t-1|t-1} \\
&= \boldsymbol{\Upsilon}_{t-1|t-1} - \mathbf{J}_{t-1}\boldsymbol{\Upsilon}_{t|t-1}\mathbf{J}_{t-1}^{\mathrm{T}}.
\end{aligned} \tag{2.54}$$

For each quantity, we have provided an alternative expression involving the cumulants of the time-updated filter distribution, Eq. 2.49. The fact that this is possible reflects the fact that future conditioning is an inversion of the same model in which the time update is a marginalization, Fig. 2.6A; and inversion requires the computation of the marginal distribution. Since running the smoother requires first running the filter, we have necessarily already computed these cumulants at this point, although we have not necessarily saved them: the smoother requires the filter only at the final time step. But we can save computation here at the price of those storage costs.

We conclude with the "backward step" of the smoother, Eq. 2.40, marginalizing out $\hat{\boldsymbol{X}}_t$. We assume that at step t, $\hat{\boldsymbol{X}}_t$ is normally distributed under the smoother distribution; and we have just shown that $\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1} \Big| \hat{\boldsymbol{X}}_t, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\Big)$ is a normal distribution about an affine function of $\hat{\boldsymbol{X}}_t$. Therefore we can apply Eq. 2.27 to compute the marginal cumulants:

Backward Step

$$\begin{aligned}
\hat{p}\Big(\hat{\boldsymbol{X}}_{t-1}\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big) &= \mathcal{N}\Big(\bar{\hat{\boldsymbol{x}}}_{t-1|T}, \boldsymbol{\Upsilon}_{t-1|T}\Big),\\
\bar{\hat{\boldsymbol{x}}}_{t-1|T} &= \bar{\hat{\boldsymbol{x}}}_{t-1|t-1} + \mathbf{J}_{t-1}\Big(\bar{\hat{\boldsymbol{x}}}_{t|T} - \bar{\hat{\boldsymbol{x}}}_{t|t-1}\Big)\\
\boldsymbol{\Upsilon}_{t-1|T} &= \mathbf{J}_{t-1}\boldsymbol{\Upsilon}_{t|T}\mathbf{J}_{t-1}^{\mathrm{T}} + \Big(\boldsymbol{\Upsilon}_{t-1|t-1} - \mathbf{J}_{t-1}\boldsymbol{\Upsilon}_{t|t-1}\mathbf{J}_{t-1}^{\mathrm{T}}\Big)\\
&= \boldsymbol{\Upsilon}_{t-1|t-1} + \mathbf{J}_{t-1}\Big(\boldsymbol{\Upsilon}_{t|T} - \boldsymbol{\Upsilon}_{t|t-1}\Big)\mathbf{J}_{t-1}^{\mathrm{T}}.
\end{aligned} \tag{2.55}$$

Let us try to interpret these equations. Notice that they relate the means and the covariances of four different recognition distributions. Essentially, we are updating the smoother (stepping backwards in time) by updating a smoother-filter disparity:

$$\Big(\bar{\hat{\boldsymbol{x}}}_{t-1|T} - \bar{\hat{\boldsymbol{x}}}_{t-1|t-1}\Big) = \mathbf{J}_{t-1}\Big(\bar{\hat{\boldsymbol{x}}}_{t|T} - \bar{\hat{\boldsymbol{x}}}_{t|t-1}\Big).$$

Both disparities measure how much an estimate changes when we do without the future observations, from t to T. The less these observations matter in estimating $\hat{\boldsymbol{X}}_t$ (i.e., the smaller the right-hand disparity), the less they matter in estimating $\hat{\boldsymbol{X}}_{t-1}$ (the smaller the left-hand disparity).

We emphasize that both filter estimates, $\bar{\hat{\boldsymbol{x}}}_{t-1|t-1}$ and $\bar{\hat{\boldsymbol{x}}}_{t|t-1}$, are made on the basis of observations only up to time $t-1$; that is, they differ only by a time update, i.e. by taking into account the state transition. Therefore, if the state transition is particularly noisy, relative to the filter's current precision, we shouldn't take $\bar{\hat{\boldsymbol{x}}}_{t|t-1}$ so seriously, and accordingly the right-hand disparity should not propagate into much of a left-hand disparity. The "gain," $\mathbf{J}_{t-1}$, should be small. On the other hand, if the state transition is particularly precise, relative to the filter's current precision, then the right-hand disparity probably reflects a problem with the filter (at this point in time), and accordingly should propagate into the left-side disparity. The gain should be large.

Apparently, then, the gain $\mathbf{J}_{t-1}$ should encode the precision of the state-transition distribution, relative to the filter distribution at time $t-1$. From its definition in Eq. 2.54 and the original definition of the gain, Eq. 2.15, we see that this is exactly what $\mathbf{J}_{t-1}$ does. It should not be suprising, then, that this gain does not depend on the future observations—indeed, it can be computed in the forward pass (Eq. 2.54)!—even though it is used to update the smoother. The gain doesn't relate the filter and smoother cumulants; it relates their disparities at two adjacent time steps. And these disparities are related to each other only insofar as time-updating the filter doesn't much damage its precision.

Similarly, taking into account the future observations shifts, or rather shrinks, the standard deviations of the recognition covariances of $\hat{\boldsymbol{X}}_{t-1}$ and $\hat{\boldsymbol{X}}_t$ by a proportional amount.

Finally, to completely characterize the recognition distribution, we also require the *cross covariance*:

$$\boldsymbol{\Upsilon}_{t,t-1|T} := \mathrm{Cov}\Big[\hat{\boldsymbol{X}}_t, \hat{\boldsymbol{X}}_{t-1}|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big].$$

Once more, we note that the relevant marginal distributions are normal, so their joint is as well, and we can apply Eq. 2.19:

$$\mathrm{Cov}\Big[\hat{\boldsymbol{X}}_t, \hat{\boldsymbol{X}}_{t-1}|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big] = \boldsymbol{\Upsilon}_{t|T}\mathbf{J}_{t-1}^{\mathrm{T}}. \tag{2.56}$$

Implementation and computational complexity. Let us consider first a detail of the implementation of the smoother. We noted above that the "future-conditioned" mean, $\tilde{\hat{\boldsymbol{X}}}_{t-1|t,T}$, is a random variable. How do we represent it? We could in theory store the numerical values of the parameters that effect the affine transformations, from Eq. 2.54 (along with, at least implicitly, the transformation itself). But in fact most of the future-conditioning step has been absorbed into the backward step, Eq. 2.55, and in practice the former amounts merely to computation of the gain, $\mathbf{J}_{t-1}$. Breaking the backward step into two pieces was for us merely a conceptual step, a way to exhibit the parallelism of the filter and smoother.

Next, consider the case where the observation space is (much) higher-dimensional than the latent space. This occurs, for example, in neural decoders, where the firing rates of hundreds of neurons are observed, but are thought to represent only some low-dimensional dynamical system. When implementing filter and smoother algorithms for such data, we would have the matrix inversions occur in the latent space.....

Sampling-based approaches

The tractability—and elegance—of the Kalman filter and RTS smoother result from the adoption of linear-Gaussian dynamics and observations. What do we do if these assumptions are inappropriate? In some cases, in may still be possible to derive closed-form update equations; but there is no reason in general why the two key operations, marginalization and especially "Bayes inversion," should have such solutions. In particular, if the source distribution in each inversion is not conjugate to the emission, i.e., if it does not under Bayes's rule yield a recognition distribution in the same family, then recursive update equations are unlikely to be derivable.

An alternative is to filter and smooth approximately with *samples*, a technique known as *particle filtering and smoothing*. We review here a simple version that corresponds closely to the exact inference algorithms just discussed (we turn to the broader picture at the end of this section). It can be summarized picturesquely in terms of one of the original applications of the Kalman filter, tracking airplanes via radar. Essentially, Kalman used the fact that airplanes do not jump around to reject (filter out) noise in the radar readings, computing the (exact) recognition distribution over the dynamical state of the airplane given all the observations (the smoother), or all until the present time (the filter)—assuming the linear-Gaussian assumptions hold. In the sampling-based approach, by contrast, one *generates 10,000 (imaginary) airplanes*, and propagates forward those that are most consistent with the observations at that time step. Then the process repeats. More precisely, one draws I samples or "particles" from the source distribution over the initial state; *weights* each particle by its "likelihood" under the observation at the initial time; then steps forward in time by drawing particles from the mixture model whose mixture weights are the (normalized) emission probabilities ("likelihoods") of the particles, and whose emission is the state-transition distribution. The smoother, for its part, works backwards through time reweighting these same samples by taking into account the future observations.

This description of the procedure may sound very different from those given above for the exact inference algorithms, but in fact the particle filter differs in only one of the four half-updates from an exact inference algorithm—filtering and smoothing *in the HMM*. The similarity to the HMM arises from the fact that sample representations of distributions can be interpreted as categorical distributions, with one category for each sample or particle. Indeed, our filter and smoother distributions will assign a "weight" to each particle, which

corresponds to the probability of that "category." Thus we can write:

$$\begin{aligned}\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_t\Big) &\approx \text{Categ}(\boldsymbol{\alpha}_t) = \sum_{i=1}^{I}\alpha_t^{(i)}\delta(\hat{\boldsymbol{X}}_t - \hat{\boldsymbol{x}}_t^{(i)}) \qquad \text{(filtering)}\\ \hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\Big) &\approx \text{Categ}(\boldsymbol{\beta}_t) = \sum_{i=1}^{I}\beta_t^{(i)}\delta(\hat{\boldsymbol{X}}_t - \hat{\boldsymbol{x}}_t^{(i)}) \qquad \text{(smoothing)}.\end{aligned} \tag{2.57}$$

Some care must be taken with this interpretation, however, because the numerical values associated with each "side of the I-sided die" will *change with every time step*. That is, we shall not simply "grid up" the state space and then ask at each time step for the probability of each of the points of this grid. This would be an inefficient sampling of the space (unless the filter distribution really is roughly uniform throughout!). Instead, we shall draw a new set of I numerical values at each step of the filter. (The smoother, on the other hand, *will* re-use these $I \times T$ samples.)

The time update, then—marginalization in Fig. 2.6A—is executed by *sampling* from the joint distribution in the graphical-model fragment, and retaining only the samples of $\hat{\boldsymbol{X}}_t$. Since the "source" distribution is categorical, the joint is a mixture model. Therefore at time t, we pick one of the I particles, $\hat{\boldsymbol{x}}_{t-1}^{(i)}$, where the probability of picking particle i is given by the corresponding "mixture weight," $\alpha_{t-1}^{(i)}$; then we use the transition probability conditioned on this sample to draw $\hat{\boldsymbol{x}}_t^{(i)}$:

Time Update

$$\begin{aligned}\hat{\boldsymbol{x}}_{t-1}^{(i)} &\xleftarrow{\text{sample}} \text{Categ}(\boldsymbol{\alpha}_{t-1})\\ \hat{\boldsymbol{x}}_t^{(i)} &\xleftarrow{\text{sample}} \hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\hat{\boldsymbol{x}}_{t-1}^{(i)}\Big).\end{aligned} \tag{2.58}$$

In practice we do this I times, so that the number of particles doesn't change from step to step, but in theory any number of samples could be chosen (perhaps based on, say, a current estimate of the variance of the distribution). Thus,

$$\hat{p}\Big(\hat{\boldsymbol{X}}_t\Big|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}\Big) \approx \text{Categ}\left(\frac{1}{I}\mathbf{1}\right) = \frac{1}{I}\sum_{i=1}^{I}\delta(\hat{\boldsymbol{X}}_t - \hat{\boldsymbol{x}}_t^{(i)}). \tag{2.59}$$

Notice that this is identical to the filter distribution in Eq. 2.57 except that the weights are the same for all our particles.

This is the only half-update in which we sample. In all the others, we simply *follow the inference procedure for an HMM*. This is possible because, as we have just seen for the filter, the recognition distributions are categorical throughout. Thus in the measurement update, we invert the mixture model in Fig. 2.6B by applying Eq. 2.6 to Eq. 2.59

Measurement Update

$$\begin{aligned}\hat{p}\left(\hat{\boldsymbol{x}}_t^{(j)}\middle|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_t\right) &\approx \frac{\frac{1}{I}\hat{p}\left(\boldsymbol{y}_t\middle|\hat{\boldsymbol{x}}_t^{(j)}\right)}{\frac{1}{I}\sum_{i=1}^{I}\hat{p}\left(\boldsymbol{y}_t\middle|\hat{\boldsymbol{x}}_t^{(i)}\right)} \\ &= \frac{\hat{p}\left(\boldsymbol{y}_t\middle|\hat{\boldsymbol{x}}_t^{(j)}\right)}{\sum_{i=1}^{I}\hat{p}\left(\boldsymbol{y}_t\middle|\hat{\boldsymbol{x}}_t^{(i)}\right)} =: \alpha_t^{(j)},\end{aligned} \tag{2.60}$$

where the definition in the final line guarantees consistency with Eq. 2.57 for all time. To keep the derivation clean, we simply assert (here and below) the support of this distribution to be the set of particles generated in the preceding time update, rather than appending a delta distribution for each particle and summing over all of them. The weights in this distribution are quite intuitive: they measure how consistent each particle is with the current observation, $\boldsymbol{y}_t$ (and then normalize). It only remains to initialize the recursion, but this is simple: the first samples (at the "first time update") are simply drawn from the source distribution over the initial state.

Our smoother will be identical to the HMM smoother. In the "future-conditioning" step, we invert the mixture model in Fig. 2.6A, again applying Eq. 2.6 but now to the filtering distribution in Eq. 2.57:

Future Conditioning

$$\hat{p}\left(\hat{\boldsymbol{x}}_{t-1}^{(j)}\middle|\hat{\boldsymbol{X}}_t,\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\right) \approx \frac{\alpha_{t-1}^{(j)}\hat{p}\left(\hat{\boldsymbol{X}}_t\middle|\hat{\boldsymbol{x}}_{t-1}^{(j)}\right)}{\sum_{i=1}^{I}\alpha_{t-1}^{(i)}\hat{p}\left(\hat{\boldsymbol{X}}_t\middle|\hat{\boldsymbol{x}}_{t-1}^{(i)}\right)}. \tag{2.61}$$

We note that this can be interpreted as another categorical distribution, although we forbear assigning a symbol to the class probabilities. Finally, in the "backward step," we marginalize the mixture model in Fig. 2.6C with Eq. 2.10, treating this future-conditioned distribution (Eq. 2.61) as the emission, and the previous smoother density (see Eq. 2.57) as the source distribution:

Backward Step

$$\hat{p}\left(\hat{\boldsymbol{x}}_{t-1}^{(j)}\middle|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\right) \approx \sum_{k=1}^{I}\beta_t^{(k)}\frac{\alpha_{t-1}^{(j)}\hat{p}\left(\hat{\boldsymbol{x}}_t^{(k)}\middle|\hat{\boldsymbol{x}}_{t-1}^{(j)}\right)}{\sum_{i=1}^{I}\alpha_{t-1}^{(i)}\hat{p}\left(\hat{\boldsymbol{x}}_t^{(k)}\middle|\hat{\boldsymbol{x}}_{t-1}^{(i)}\right)} =: \beta_{t-1}^{(j)}. \tag{2.62}$$

The definition in the final line guarantees that the smoother distribution satisfies Eq. 2.57 for all time, as long as it is initialized properly. And indeed, it is also obvious from Eq. 2.57 that $\boldsymbol{\beta}_T = \boldsymbol{\alpha}_T$. Notice that this is the same set of particles used to represent the filter distribution! but, moving backward from time T, the weights on those particles ($\boldsymbol{\beta}_t$) will (typically) depart from the weights of the filter ($\boldsymbol{\alpha}_t$).

A more general approach to particle filtering [**?**]....

2.2.3 Sparse dynamical models

....

Chapter 3

Undirected Generative Models

In Chapter 2, we considered modeling tasks in which we begin with some knowledge or intuition about the conditional probability of certain variables, given certain others. After assembling distributions for all the relevant variables, we can construct a joint distribution out of their product. Now we consider modeling tasks in which we begin with some intuitions or knowledge only about how "stable" certain configurations of variables are—that is, with unnormalized probability distributions. [[It might seem that we can just normalize all these, turn them into conditionals, and then make a directed graphical model.... Certain UGMs that can't be turned into DGMs, e.g. the square.... In practice, we do the reverse....]]

.... We saw in Chapter 2 that inference in directed graphical models is essentially some kind of more or less complex application of Bayes's rule. But abstracting away from the precise meaning of the probability distributions in Eq. **??**, we see that the fundamental operations are multiplication, marginalization, and normalization, and these carry over to the undirected setting.... [[This will be relevant for our investigation into probabilistic computation in the brain....]]

3.1 The exponential-family harmonium

In Chapter 2, we encountered a direct trade-off between the expressivity of the model emission distribution, $\hat{p}\left(\hat{\boldsymbol{Y}} \middle| \hat{\boldsymbol{X}}; \boldsymbol{\theta}\right)$, and the model recognition distribution, $\hat{p}\left(\hat{\boldsymbol{X}} \middle| \hat{\boldsymbol{Y}}; \boldsymbol{\theta}\right)$, imposed by Bayes's theorem. In particular, applying Bayes's theorem requires integrating the product of the emission and source ($\hat{p}\left(\hat{\boldsymbol{X}}; \boldsymbol{\theta}\right)$) densities across all configurations of the source variables, $\hat{\boldsymbol{X}}$. For continuous-valued $\hat{\boldsymbol{X}}$, this integral is tractable only for specially selected source and emission densities. For discrete-valued $\hat{\boldsymbol{X}}$, the integral becomes a sum, which is only computationally feasible for low-dimensional $\hat{\boldsymbol{X}}$: the number of summands is exponential in $\left|\hat{\boldsymbol{X}}\right|$.

Suppose instead, then, we simply declared at the outset our two (so far) desiderata: easily computable emission *and* recognition distributions. Of course, not every pair of such distributions will be compatible, but perhaps if we start with some very general form for these distributions, we can subsequently determine what restrictions will be required for their consistency. In so doing, we shall have derived a rather general undirected graphical model known as the *exponential-family harmonium* [27]. In fact, the EFH was derived as a generalization of the famous *restricted Boltzmann machine* [23], but we shall approach from the other end and present the RBM as a special case of the EFH.

Deriving the joint from two coupled, exponential-family conditionals. We shall not assume the emission and recognition distributions fully general, but that the are in exponential families. Note that this need not be the *same* exponential family; indeed, the several elements of (e.g.) $\hat{\boldsymbol{X}}$ need not even belong to the same one. Nevertheless, we can specify that the two distributions have the forms

$$\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) = h(\hat{\boldsymbol{X}})\exp\Big\{\boldsymbol{\eta}(\hat{\boldsymbol{Y}})^{\mathrm{T}}\boldsymbol{T}(\hat{\boldsymbol{X}}) - A(\boldsymbol{\eta}(\hat{\boldsymbol{Y}}))\Big\},$$
$$\hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = k(\hat{\boldsymbol{Y}})\exp\Big\{\boldsymbol{\zeta}(\hat{\boldsymbol{X}})^{\mathrm{T}}\boldsymbol{U}(\hat{\boldsymbol{Y}}) - B(\boldsymbol{\zeta}(\hat{\boldsymbol{X}}))\Big\}.$$

Thus, (functions of) $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$ interact with each other only through an inner product.

Now, the ratio of the conditionals is also the ratio of the marginals,

$$\frac{\hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)}{\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)} = \frac{\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)}{\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)} = \frac{k(\hat{\boldsymbol{Y}})}{h(\hat{\boldsymbol{X}})}\frac{\exp\Big\{A(\boldsymbol{\eta}(\hat{\boldsymbol{Y}}))\Big\}}{\exp\Big\{B(\boldsymbol{\zeta}(\hat{\boldsymbol{X}}))\Big\}}\exp\Big\{\boldsymbol{\zeta}(\hat{\boldsymbol{X}})^{\mathrm{T}}\boldsymbol{U}(\hat{\boldsymbol{Y}}) - \boldsymbol{\eta}(\hat{\boldsymbol{Y}})^{\mathrm{T}}\boldsymbol{T}(\hat{\boldsymbol{X}})\Big\},$$

but we know an additional fact about this ratio: it must factor entirely into pieces that refer to at most one of $\hat{\boldsymbol{X}}$ or $\hat{\boldsymbol{Y}}$. The first two (rational) factors look fine, but the third term requires that

$$\boldsymbol{\zeta}(\hat{\boldsymbol{X}})^{\mathrm{T}}\boldsymbol{U}(\hat{\boldsymbol{Y}}) - \boldsymbol{\eta}(\hat{\boldsymbol{Y}})^{\mathrm{T}}\boldsymbol{T}(\hat{\boldsymbol{X}}) = \mu(\hat{\boldsymbol{X}}) - \nu(\hat{\boldsymbol{Y}}), \tag{3.1}$$

for some functions μ and ν. It can be shown (see the proof below) that under some mild conditions, this requires each distribution's sufficient statistics to be an affine function of the other distribution's natural parameters, with a shared, albeit transposed, linear transformation $\mathbf{W}_{\hat{y}\hat{x}}$:

$$\boldsymbol{\eta}(\hat{\boldsymbol{Y}}) = \boldsymbol{b}_{\hat{x}} + \mathbf{W}_{\hat{y}\hat{x}}\boldsymbol{U}(\hat{\boldsymbol{Y}})$$
$$\boldsymbol{\zeta}(\hat{\boldsymbol{X}}) = \boldsymbol{b}_{\hat{y}} + \mathbf{W}_{\hat{y}\hat{x}}^{\mathrm{T}}\boldsymbol{T}(\hat{\boldsymbol{X}}).$$

Therefore, the marginal distributions are (up to the proportionality constants)

$$\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) \propto h(\hat{\boldsymbol{X}})\exp\Big\{\boldsymbol{b}_{\hat{x}}{}^{\mathrm{T}}\boldsymbol{T}(\hat{\boldsymbol{X}}) + B(\boldsymbol{\zeta}(\hat{\boldsymbol{X}}))\Big\},$$
$$\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) \propto k(\hat{\boldsymbol{Y}})\exp\Big\{\boldsymbol{b}_{\hat{y}}{}^{\mathrm{T}}\boldsymbol{U}(\hat{\boldsymbol{Y}}) + A(\boldsymbol{\eta}(\hat{\boldsymbol{Y}}))\Big\}.$$

and the conditional distributions are

$$\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) = h(\hat{\boldsymbol{X}})\exp\Big\{\Big(\boldsymbol{b}_{\hat{x}} + \mathbf{W}_{\hat{y}\hat{x}}\boldsymbol{U}(\hat{\boldsymbol{Y}})\Big)^{\mathrm{T}}\boldsymbol{T}(\hat{\boldsymbol{X}}) - A(\boldsymbol{\eta}(\hat{\boldsymbol{Y}}))\Big\},$$
$$\hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = k(\hat{\boldsymbol{Y}})\exp\Big\{\Big(\boldsymbol{b}_{\hat{y}} + \mathbf{W}_{\hat{y}\hat{x}}^{\mathrm{T}}\boldsymbol{T}(\hat{\boldsymbol{X}})\Big)^{\mathrm{T}}\boldsymbol{U}(\hat{\boldsymbol{Y}}) - B(\boldsymbol{\zeta}(\hat{\boldsymbol{X}}))\Big\}.$$

Multiplying a conditional by the appropriate marginal yields the joint distribution:

$$\begin{aligned}\hat{p}\Big(\hat{\boldsymbol{X}},\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) &= \hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)\\ &\propto h(\hat{\boldsymbol{X}})k(\hat{\boldsymbol{Y}})\exp\Big\{\boldsymbol{b}_{\hat{y}}{}^{\mathrm{T}}\boldsymbol{U}(\hat{\boldsymbol{Y}}) + \boldsymbol{b}_{\hat{x}}{}^{\mathrm{T}}\boldsymbol{T}(\hat{\boldsymbol{X}}) + \boldsymbol{U}(\hat{\boldsymbol{Y}})^{\mathrm{T}}\mathbf{W}_{\hat{y}\hat{x}}^{\mathrm{T}}\boldsymbol{T}(\hat{\boldsymbol{X}})\Big\}.\end{aligned}$$

Thus the joint takes the form of a Boltzmann distribution with negative energy

$$-E(\boldsymbol{\theta}) = \boldsymbol{b}_{\hat{y}}{}^{\mathrm{T}}\boldsymbol{U}(\hat{\boldsymbol{Y}}) + \boldsymbol{b}_{\hat{x}}{}^{\mathrm{T}}\boldsymbol{T}(\hat{\boldsymbol{X}}) + \boldsymbol{U}(\hat{\boldsymbol{Y}})^{\mathrm{T}}\mathbf{W}_{\hat{y}\hat{x}}^{\mathrm{T}}\boldsymbol{T}(\hat{\boldsymbol{X}}) + \log\Big(h(\hat{\boldsymbol{X}})k(\hat{\boldsymbol{Y}})\Big).$$

The price of trivial inference. We can now reckon the cost at which our closed-form recognition distribution was bought. We have traded an intractable recognition-distribution normalizer for an intractable joint-distribution normalizer. The normalizer for the marginal distribution $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$ is still intractable, as it is for many directed models, but now so is the normalizer for the source distribution, $\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)$.

...

Enforcing consistency between exponential-family emission and recognition distributions. We saw above that when the emission and recognition distributions are both in exponential families, the natural parameters are constrained by Eq. 3.1. To simplify the presentation, we repeat the constraint here (with the vector-valued functions named alphabetically):

$$\mu(\hat{\boldsymbol{x}}) - \nu(\hat{\boldsymbol{y}}) = \boldsymbol{\gamma}(\hat{\boldsymbol{x}})^{\mathrm{T}}\boldsymbol{\delta}(\hat{\boldsymbol{y}}) - \boldsymbol{\beta}(\hat{\boldsymbol{y}})^{\mathrm{T}}\boldsymbol{\alpha}(\hat{\boldsymbol{x}}). \tag{3.2}$$

It is intuitive that this equation constrains the natural parameters (here, $\boldsymbol{\gamma}(\hat{\boldsymbol{x}})$ and $\boldsymbol{\beta}(\hat{\boldsymbol{y}})$): no $\hat{\boldsymbol{x}}$-$\hat{\boldsymbol{y}}$ interaction terms appear on the left-hand side, so those generated on the right must cancel. This is particularly restrictive since the interactions are created only through inner products. For example, if $\delta(\hat{\boldsymbol{y}})$ contains only terms quadratic in the elements of $\hat{\boldsymbol{y}}$, then $\boldsymbol{\beta}(\hat{\boldsymbol{y}})$ must contain such terms as well, in order to cancel them (except in the trivial case where $\boldsymbol{\gamma}(\hat{\boldsymbol{x}})$ is constant).

Let all the functions be polynomials in $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ of maximum degree D, and define the monomial bases

$$\begin{aligned}\boldsymbol{v}_{\hat{y}} &:= \left[\hat{y}_1, \hat{y}_2, \ldots, {\hat{y}_1}^2, \hat{y}_1\hat{y}_2, \hat{y}_1\hat{y}_3, \ldots, {\hat{y}_K}^D\right]^{\mathrm{T}} \\ \boldsymbol{v}_{\hat{x}} &:= \left[\hat{x}_1, \hat{x}_2, \ldots, {\hat{x}_1}^2, \hat{x}_1\hat{x}_2, \hat{x}_1\hat{x}_3, \ldots, {\hat{x}_K}^D\right]^{\mathrm{T}}.\end{aligned}$$

(Notice that we have omitted the constants from these bases.) For appropriately shaped matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$, vectors $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d})$, and constant (k), Eq. 3.2 can then be written

$$\begin{aligned}\boldsymbol{m}^{\mathrm{T}}\boldsymbol{v}_{\hat{x}} - \boldsymbol{n}^{\mathrm{T}}\boldsymbol{v}_{\hat{y}} + k &= (\boldsymbol{c} + \mathbf{C}\boldsymbol{v}_{\hat{x}})^{\mathrm{T}}\big(\boldsymbol{d} + \mathbf{D}\boldsymbol{v}_{\hat{y}}\big) - \big(\boldsymbol{b} + \mathbf{B}\boldsymbol{v}_{\hat{y}}\big)^{\mathrm{T}}(\boldsymbol{a} + \mathbf{A}\boldsymbol{v}_{\hat{x}}) \\ &= {\boldsymbol{v}_{\hat{x}}}^{\mathrm{T}}\big(\mathbf{C}^{\mathrm{T}}\mathbf{D} - \mathbf{A}^{\mathrm{T}}\mathbf{B}\big)\boldsymbol{v}_{\hat{y}} + {\boldsymbol{v}_{\hat{x}}}^{\mathrm{T}}\big(\mathbf{C}^{\mathrm{T}}\boldsymbol{d} - \mathbf{A}^{\mathrm{T}}\boldsymbol{b}\big) + \big(\boldsymbol{c}^{\mathrm{T}}\mathbf{D} - \boldsymbol{a}^{\mathrm{T}}\mathbf{B}\big)\boldsymbol{v}_{\hat{y}} + \big(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{d} - \boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}\big).\end{aligned}$$

This equation must hold for all values of $\boldsymbol{v}_{\hat{x}}$ and $\boldsymbol{v}_{\hat{y}}$. Therefore,

$$\begin{aligned}k &= \boldsymbol{c}^{\mathrm{T}}\boldsymbol{d} - \boldsymbol{a}^{\mathrm{T}}\boldsymbol{b} \\ \boldsymbol{m} &= \mathbf{C}^{\mathrm{T}}\boldsymbol{d} - \mathbf{A}^{\mathrm{T}}\boldsymbol{b} \\ -\boldsymbol{n}^{\mathrm{T}} &= \boldsymbol{c}^{\mathrm{T}}\mathbf{D} - \boldsymbol{a}^{\mathrm{T}}\mathbf{B} \\ 0 &= \mathbf{C}^{\mathrm{T}}\mathbf{D} - \mathbf{A}^{\mathrm{T}}\mathbf{B}.\end{aligned} \tag{3.3}$$

We shall only make use of the last of these, Eq. 3.3.

Now assume $\mathbf{A}$ and $\mathbf{D}$ are "fat"—that is, $D \geq K$: the monomial bases $\boldsymbol{v}_{\hat{x}}$ and $\boldsymbol{v}_{\hat{y}}$ have at least as many elements as the vector-valued functions $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ (resp.)—with linearly independent columns. Then there exists a (tall) right pseudo-inverse for $\mathbf{A}$, call it $\mathbf{A}^{\dagger}$, such that $\mathbf{A}\mathbf{A}^{\dagger} = \mathbf{I}$; and a (tall) right pseudo-inverse for $\mathbf{D}$, call it $\mathbf{D}^{\dagger}$, such that $\mathbf{D}\mathbf{D}^{\dagger} = \mathbf{I}$. It

follows immediately from the last of Eq. 3.3 that

$$\begin{aligned} & \mathbf{C}^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}}\mathbf{B}\mathbf{D}^{\dagger}, \qquad \mathbf{B} = \left(\mathbf{C}\mathbf{A}^{\dagger}\right)^{\mathrm{T}}\mathbf{D} \\ \implies & \left(\mathbf{C}\mathbf{A}^{\dagger}\right)^{\mathrm{T}} = \mathbf{B}\mathbf{D}^{\dagger} =: \mathbf{W} \\ \implies & \mathbf{C}^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}}\mathbf{W}, \qquad \mathbf{B} = \mathbf{W}\mathbf{D}, \end{aligned} \tag{3.4}$$

where on the second line we have defined a new matrix $\mathbf{W}$. This allows us to rewrite the functions $\boldsymbol{\gamma}(\hat{\boldsymbol{x}})$ and $\boldsymbol{\beta}(\hat{\boldsymbol{y}})$ in terms of $\boldsymbol{\alpha}(\hat{\boldsymbol{x}})$ and $\boldsymbol{\delta}(\hat{\boldsymbol{y}})$ (resp.):

$$\begin{aligned} \boldsymbol{\gamma}(\hat{\boldsymbol{x}}) &= \mathbf{C}\boldsymbol{v}_{\hat{x}} + \boldsymbol{c} \\ &= \mathbf{W}^{\mathrm{T}}\mathbf{A}\boldsymbol{v}_{\hat{x}} + \boldsymbol{c} \\ &= \mathbf{W}^{\mathrm{T}}(\mathbf{A}\boldsymbol{v}_{\hat{x}} + \boldsymbol{a}) + \left(\boldsymbol{c} - \mathbf{W}^{\mathrm{T}}\boldsymbol{a}\right) \\ &= \mathbf{W}^{\mathrm{T}}\boldsymbol{\alpha}(\hat{\boldsymbol{x}}) + \left(\boldsymbol{c} - \mathbf{W}^{\mathrm{T}}\boldsymbol{a}\right) \\ \boldsymbol{\beta}(\hat{\boldsymbol{y}}) &= \mathbf{B}\boldsymbol{v}_{\hat{y}} + \boldsymbol{b} \\ &= \mathbf{W}\mathbf{D}\boldsymbol{v}_{\hat{y}} + \boldsymbol{b} \\ &= \mathbf{W}\left(\mathbf{D}\boldsymbol{v}_{\hat{y}} + \boldsymbol{d}\right) + (\boldsymbol{b} - \mathbf{W}\boldsymbol{d}) \\ &= \mathbf{W}\boldsymbol{\delta}(\hat{\boldsymbol{y}}) + (\boldsymbol{b} - \mathbf{W}\boldsymbol{d}). \end{aligned} \tag{3.5}$$

In a word, $\boldsymbol{\gamma}(\hat{\boldsymbol{x}})$ is an affine function of $\boldsymbol{\alpha}(\hat{\boldsymbol{x}})$, and $\boldsymbol{\beta}(\hat{\boldsymbol{y}})$ is an affine function of $\boldsymbol{\delta}(\hat{\boldsymbol{y}})$; and the linear transformations are transposes of each other.

3.2 Deep belief networks

3.3 The Helmholtz machine

3.4 Recurrent EFHs

3.4.1 The recurrent temporal RBM

3.4.2 The recurrent EFH

3.5 General inference algorithms

[[So far, inference has looked like a (possibly iterative) application of Bayes's theorem. The most complicated structure we've considered is the binary tree underlying the HMM and the state-space models. Now we'd like to generalize to more complicated directed and undirected graphs.

The basic intuition is that we can construct efficient algorithms for trees and tree-like graphs. Therefore, the basis of our derivation of the very general junction-tree algorithm will be to convert non-tree graphs into something tree-like. We may have to bite the bullet and accept large cliques, which are bad from a computation perspective.... The alternatives are approximate inference (loopy BP, variational inference, sampling methods).

At the heart of inference is normalization. This is the "hard" part of Bayes's rule, because it may involve an intractable integral; or it may involve summing over all the configurations of the random variables, the number of the former being exponential in the number of the latter.

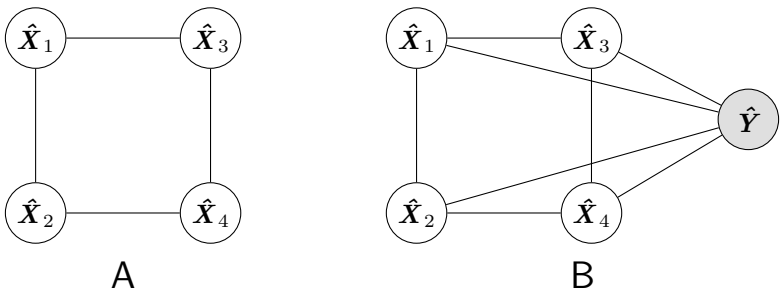


Figure 3.1: **Alternative interpretation of undirected model.** The unnormalized distribution represented by the product of potentials in (A) corresponds to a slice through the normalized distribution represented by the graph in (B).

The other setting in which we had to compute normalizers was undirected graphical models, since the product of the potential functions on an undirected graph is an *unnormalized* distribution. But this too can be assimilated to inference with Bayes's theorem. To see this, consider the undirected graphical model in Fig. 3.1A, for which the joint distribution is

$$\hat{p}(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2, \hat{\boldsymbol{X}}_3, \hat{\boldsymbol{X}}_4) \propto \psi_A(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2)\psi_B(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_3)\psi_C(\hat{\boldsymbol{X}}_2, \hat{\boldsymbol{X}}_4)\psi_D(\hat{\boldsymbol{X}}_3, \hat{\boldsymbol{X}}_4).$$

The independence statements asserted by this graph—e.g., $\hat{\boldsymbol{X}}_1 \perp\!\!\!\perp \hat{\boldsymbol{X}}_4 | \hat{\boldsymbol{X}}_2, \hat{\boldsymbol{X}}_3$—follow from the usual graph-separation criterion.

Now consider the graphical model in Fig. 3.1B, which we assert to be *normalized*. Thus

$$\check{p}(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2, \hat{\boldsymbol{X}}_3, \hat{\boldsymbol{X}}_4, \hat{\boldsymbol{Y}}) = \check{\psi}_A(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2, \hat{\boldsymbol{Y}})\check{\psi}_B(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_3, \hat{\boldsymbol{Y}})\check{\psi}_C(\hat{\boldsymbol{X}}_2, \hat{\boldsymbol{X}}_4, \hat{\boldsymbol{Y}})\check{\psi}_D(\hat{\boldsymbol{X}}_3, \hat{\boldsymbol{X}}_4, \hat{\boldsymbol{Y}}).$$

In this graph, $\hat{\boldsymbol{X}}_1$ and $\hat{\boldsymbol{X}}_4$ are no longer independent conditioned on $\hat{\boldsymbol{X}}_2, \hat{\boldsymbol{X}}_3$, since there is a connecting path through $\hat{\boldsymbol{Y}}$. But conditioning on $\hat{\boldsymbol{Y}}$ clearly restores all of the independence statements of Fig. 3.1A. Therefore if, for a particular value $\hat{\boldsymbol{y}}$ of $\hat{\boldsymbol{Y}}$, we define

$$\begin{aligned}
\check{\psi}_A(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2, \hat{\boldsymbol{y}}) &:= \psi_A(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2) & \check{\psi}_B(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2, \hat{\boldsymbol{y}}) &:= \psi_B(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_3) \\
\check{\psi}_C(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2, \hat{\boldsymbol{y}}) &:= \psi_C(\hat{\boldsymbol{X}}_2, \hat{\boldsymbol{X}}_4) & \check{\psi}_D(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2, \hat{\boldsymbol{y}}) &:= \psi_D(\hat{\boldsymbol{X}}_3, \hat{\boldsymbol{X}}_4),
\end{aligned}$$

then

$$\begin{aligned}
\check{p}(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2, \hat{\boldsymbol{X}}_3, \hat{\boldsymbol{X}}_4, \hat{\boldsymbol{y}}) &= \psi_A(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2)\psi_B(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_3)\psi_C(\hat{\boldsymbol{X}}_2, \hat{\boldsymbol{X}}_4)\psi_D(\hat{\boldsymbol{X}}_3, \hat{\boldsymbol{X}}_4) \\
\implies \check{p}(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2, \hat{\boldsymbol{X}}_3, \hat{\boldsymbol{X}}_4 | \hat{\boldsymbol{y}}) &= \frac{1}{\check{p}(\hat{\boldsymbol{Y}})}\psi_A(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2)\psi_B(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_3)\psi_C(\hat{\boldsymbol{X}}_2, \hat{\boldsymbol{X}}_4)\psi_D(\hat{\boldsymbol{X}}_3, \hat{\boldsymbol{X}}_4) \\
&= \hat{p}(\hat{\boldsymbol{X}}_1, \hat{\boldsymbol{X}}_2, \hat{\boldsymbol{X}}_3, \hat{\boldsymbol{X}}_4).
\end{aligned}$$

The last line follows from summing both sides over all configurations of $\hat{\boldsymbol{X}}$. In fine, the missing normalizer is $\check{p}(\hat{\boldsymbol{Y}})$.

More generally, the product of potentials for any undirected graphical model with nodes $\hat{\boldsymbol{X}}_1, \ldots, \hat{\boldsymbol{X}}_N$ can be interpreted as a *slice* through some normalized distribution, $\check{p}(\hat{\boldsymbol{X}}_1, \ldots, \hat{\boldsymbol{X}}_N, \hat{\boldsymbol{y}})$, where the auxiliary random variable $\hat{\boldsymbol{Y}}$ did not occur in the original graph. Under this interpretation, computing the partition function is equivalent to computing the marginal probability of $\hat{\boldsymbol{y}}$—another instance of inference with Bayes's theorem.

]]

Part II

Learning

[[The four combinations of unsupervised, supervised, discriminative, and generative.... The four corresponding graphical models....]]

Chapter 4

A Mathematical Framework for Learning

4.1 Learning as optimization

What does it mean for a machine, biological or artificial, to "learn"? Vaguely speaking, it means incrementally changing the values of "parameters" so as to improve performance on a particular task. These parameters may be the weights and biases of an artificial neural network, the statistical parameters (means, variances, etc.) of a graphical model, or the synaptic strengths of a biological neural network. The task may be to build a map from inputs to outputs (e.g., from photographs to labels of the objects in the photos), to be able to generate samples from a target probability distribution, or to maximize rewards in some environment.

Why do we ask only for "good" rather than "correct" performance? In a very broad sense, statistical learning problems arise when no single solution is determined by the data. The solution may be underdetermined or overdetermined (see Fig. **??**), but in either case the resolution is to give up "right or wrong" in favor of "better or worse." For example, in the case of overdetermined problems, we specify *how* wrong a solution is, by assigning a "cost" to mismatches that (typically) smoothly increases with distance from the correct solution; and then attempt to minimize this cost or *loss* ($\mathcal{L}$). Thus, learning problems are typically *optimization* problems. *loss function*

More concretely, the networks we have considered in previous chapters are described by equations that determine their behavior up to the assignment of actual numbers to their parameters ($\boldsymbol{\theta}$). For some setting of the parameters, the loss is minimized (or *objective* maximized) and the task achieved—that is, as well as it can be by the machine in question. To find this setting, one applies standard tools from calculus: typically, one considers the derivative (gradient) of the loss function with respect to the parameters. The global minimum of the loss function is located where the derivative is zero (although not necessarily conversely), so ideally one can simply set $\mathrm{d}\mathcal{L}/\,\mathrm{d}\boldsymbol{\theta}$ to zero and solve for the parameters. The ideal is rarely attained in practice because the resulting equations are too difficult to be solved in closed form. Alternatively, then, a local, incremental approach is employed: parameters are changed proportional to the local gradient; that is, the algorithm walks downhill along the loss's surface in the space of parameters. It is the incremental character of this *gradient descent*, and the increase in performance that accompanies it, that gives parameter acquisition the character of "learning." Of course, gradient descent generally will *gradient descent*

find only those minima *local* to the starting point. To some extent, this weakness can be compensated for by re-running the algorithm multiple times from randomly chosen initial parameter settings.

local minimum

Ideally, the machine should function well (produce small losses) for all input/output pairs (supervised learning) or all observations (unsupervised learning)—collectively, "the data"—but these data may make conflicting demands. How are they to be adjudicated? Generally, they are weighted by the frequency of their occurrence; that is, the gradient of the loss function is evaluated under the data, and these gradients averaged. Here a choice presents itself: how many of the data should be used for this average before a step is taken? It might seem necessary to use all of them, but in fact it is neither necessary nor optimal: although subsets of the data are generally less representative of the true distribution (and therefore the true loss that will accrue on future, "test" data), computing the average of $\mathrm{d}\mathcal{L}/\,\mathrm{d}\boldsymbol{\theta}$ on a subset (or "batch") will be less time consuming. Furthermore, even if all the data are used, the algorithm will not (except in very special cases) reach the local minimum in one step, so this time cost accumulates. Essentially, one takes better steps at the price of making them more slowly; or, at the other extreme (one datum per step), takes very badly chosen steps very quickly. The optimal batch size, which may depend on details of the implementation (e.g., matrix multiplications vs. loops) and hardware (e.g., GPUs), will generally fall between the extremes. Because of the randomness introduced by sub-sampling, this procedure is known as *stochastic gradient descent*—although the "full" data distribution itself consists of samples anyway: in some sense all gradient descent under sample distributions is stochastic.

stochastic gradient descent

Stochastic gradient descent is also motivated from another perspective: For certain machines, it may be necessary to process the data as they come in—e.g. if, for whatever reason, it is impossible to store them. These "online" algorithms are in some sense more biologically plausible, since it seems unlikely that the brain stores multiple input/output pairs before changing synaptic weights. For the same reason, we are interested in such algorithms even when the gradient-zeroing equation *can* be solved analytically. Below we consider algorithms that rely on analytical solutions, batch gradient descent, and online gradient descent; but it is important to remember that those of the first type can usually be converted into those of the second and third types when required.

online vs. batch algorithms

Overfitting and regularization. Any of these methods can fail, for the various reasons just adduced, to learn the data they have been trained on, but in fact they can succeed at this and still fail to have "learned" more generally. In particular, "*overfitting*" of training data will create a model too brittle to accommodate new (i.e., test) data. The intuition behind overfitting can be brought out forcefully by imagining a machine so powerful (parameter-rich) that it can memorize all the training data ever given to it: it simply stores each input/output pair (or each input, for unsupervised learning) in its "memory." This machine will commit no errors on the training data, and only errors on the test data. Whether or not the machine fails to "generalize" in this sense is related to the ratio of training data to parameters: roughly, one wants more of the former than the latter.

overfitting

insert classic figure of high-order polynomial to noisy linear data

The obvious and seemingly straightforward remedy for overfitting, then, is simply to make sure this ratio is large enough. Unfortunately, it is often difficult to know *a priori* what "large enough" is. Consider, for example, a supervised learning task for a discriminative model with vector inputs, like a multiple linear regression. Naïvely, each element of that vector deserves its own parameter. But one input element might be correlated strongly

with another input element, or only weakly with the output, in both of which cases it deserves "less than one" parameter. When the input dependencies are linear (i.e., they are first-order correlations), the dimensionality of the data—and therefore the number of model parameters—can be reduced ahead of time by simple linear transformation (multiplication by a fat matrix). But generally, unraveling these dependencies is itself a formidable learning task—in fact, it is normally a large part of what we wanted the machine to learn for us in the first place.

One hypothetical way to handle our general uncertaintly about the ratio of information in the data and the parameters would be to train a model with a rather large number and test for failure to generalize on held-out, "validation" data. In the case of failure, we could eliminate some of the parameters and repeat the process. But which parameters should we eliminate?

If we think of this "elimination" as setting to zero, a subtler alternative suggests itself. Rather than simply removing parameters, one can instead "encourage" parameters to be zero. Essentially, one adds another demand to the objective of learning the training data: parameters are penalized for being far from zero, as well as for yielding bad performance on the training data. Parameters that have little effect on the latter penalty will not be able to offset the former penalty, and consequently will be "pulled" closer to zero. This automates and softens the elimination of parameters, although it does not by itself obviate the need for an iterative validation scheme: We still need to decide how much "encouragement" to give each parameter, that is, the relative strength of the two penalties. This is often set with a single scalar hyperparameter, λ, found by sweeping across possible values, training the model, and testing on the validation data.

regularization

The probabilistic interpretation of the loss. Adding demands to an objective function corresponds, as usual, to the method of Lagrange multipliers, with multiplier λ. But it also has a statistical interpretation: the parameters have a prior distribution whose mode is zero. Something like the variance of that distribution roughly corresponds to (the inverse of) the Lagrange multiplier—broader distributions demand less strenuously that the parameters be zero. The family of the distribution corresponds to the form of the penalty function: Gaussian for squared deviation from zero, Laplace for absolute deviation, etc. We shall see examples of this "regularization" in the following chapters.

Perhaps unsurprisingly, the original loss (before the addition of Lagrange terms) can likewise be redescribed in terms of probabilities, and indeed the entire loss in terms of a joint distribution. This may not seem particularly compelling until we are faced with the problem of coming up with a loss function. In some cases, good losses are obvious enough—when positive and negative (real-valued) deviations are equally undesirable, a sensible loss is the average *squared* error; in others, perhaps less so. Consider, for example, trying to predict the running speed (not velocity) of an animal often at rest. Squared error, under which impossible "negative speeds" are penalized just like erroneous—but possible—positive speeds, is clearly problematic.

We shall see shortly how probabilities naturally give rise to losses. Here we note simply that losses generically range across all real numbers and should be decreased by learning, whereas probabilities must be positive real numbers and (intuitively) should be increased by learning. Therefore the natural transformation from probabilities to losses is the *negative logarithm*.

The data distribution and the model distribution. In Chapters 2 and 3, we discussed several probabilistic models, that is, particular families of joint distributions of random variables. Let us refer to them simply as $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$, not yet specifying anything about the random variables $\hat{\boldsymbol{Y}}$. Recall that which distribution, as opposed to which family of distributions, a particular probabilistic model corresponds to, is determined by the numerical values of its parameters, $\boldsymbol{\theta}$. For any particular setting of $\boldsymbol{\theta}$, we shall call $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$ *the model distribution*. We also saw how to make inferences under such models: to compute marginal or conditional distributions over some subset of variables. But for these inferences to be interesting or useful, the model must be a good model of *something*.

the model distribution

In the general setting that we now consider, we begin with a set of numerical samples from "the world." These could be readings from instruments, or tabulations made by humans, or etc. The key assumption is that these samples come from a distribution, $p(\boldsymbol{Y})$, which we shall call *the data distribution*. We never have direct access to this distribution, only the samples—indeed, in contrast to the model distribution, we do not even assume that it has a parametric form.[1] But with the notion of a data distribution, we can now make more precise what "learning" means for probabilistic models: Learning is the process of adjusting the parameters, $\boldsymbol{\theta}$, so as to make the model distribution, $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$, more like the data distribution, $p(\boldsymbol{Y})$.

the data distribution

Ideally, the learning algorithm will ultimately bring about the equality of model and data distributions, $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) = p(\boldsymbol{Y})$, but notice that there are two distinct causes of failure: inadequacy of the algorithm, and inadequacy of the model. The second failure mode occurs when the data distribution $p(\boldsymbol{Y})$ is not a member of the family of distributions specified by the model; that is, when there is no setting of $\boldsymbol{\theta}$ for which the equality holds. This underscores the importance of getting the model right, or at least close to right, or again flexible enough to accommodate a very broad family of distributions. On the other hand, these desiderata can conflict with the desideratum of an efficacious learning algorithm: more expressive models generally require more approximations during inference and learning. In other words, minimizing the damage from the two failure modes itself represents a kind of (meta)optimization. We explore this trade-off throughout this part of the book, generally moving from less expressive models with exact inference and learning algorithms to more expressive models which require approximations.

In many texts, the distinction between p and $\hat{p}$ is not made, the model and data distributions are conflated, and one takes the goal of learning to be simply acquisition of the true parameters of the true model of the data. In this case, one makes reference only to a distribution called (say) $p\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$—in our view, a kind of chimera between the model and data distributions. In contrast, in this textbook, we insist on distinguishing the two—first and foremost because this is a more felicitous description of the actual position of our algorithm, and our nervous system, *vis-à-vis* "the world": Except in special circumstances, it is not likely that our models—still less our brains—are recapitulating the physical laws that are ultimately responsible for the data. This distinction also allows for using "noise" in our models to accommodate mismatches between the model and the data, without committing us to a belief in randomness inherent in the data themselves.

[1] We take up in Section **??** the possibility of dispensing with equations for the model distribution, as well.

4.2 Minimizing relative entropy and maximizing likelihood.

So we have specified the goal of learning, $\hat{p}\left(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\right) = p(\boldsymbol{Y})$, but on the other hand conceded that it may not be achievable. What we need, given the view just sketched of learning as optimization, is a notion of distance from this goal. Or in other words, if learning is "to make the model distribution, $\hat{p}\left(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\right)$, more like the data distribution, $p(\boldsymbol{Y})$," we need to operationalize "more like." To do so, we take a somewhat informal detour through elementary information theory.

A guessing game. Suppose we play a game in which I hide a marble under one of four cups, and you have to determine which by asking a series of yes/no questions. Every time we play, I select the destination cup according to the probabilities p which, suppose, are

$$p(\boldsymbol{Y}) = \begin{array}{|c|c|c|c|} \multicolumn{1}{c}{\text{A}} & \multicolumn{1}{c}{\text{B}} & \multicolumn{1}{c}{\text{C}} & \multicolumn{1}{c}{\text{D}} \\ \hline 1/2 & 1/4 & 1/8 & 1/8 \\ \hline \end{array} \tag{4.1}$$

and which you know. Your strategy is to partition the probability space into halves recursively. (Of course, this may not be feasible—e.g., if cup A had probability 1/3—but we will handle such cases later. We will also address below whether your strategy is any good.) Thus, each of your questions eliminates half the probability space, and it takes n questions to reduce the space to $(\frac{1}{2})^n$ of its original size. Put the other way around, to reduce the space to fraction q takes $-\log_2 q$ questions.

flow chart showing guessing outcomes

How many questions will it take on average (across all games)? Here, for example, you would first ask, "Is it cup A?" Half the time you would be right, so half the time you would locate the marble in one guess ((1/2)(1)). If not, you would ask, "Is it cup B?" in which case you would again be right half the time you asked the question—i.e., half of the half of the time you played the game, at which point you would have asked two questions ((1/4)(2)). Finally, in the 1/4 of the games that you received two "no"s, you would ask "Is it cup C?" and from either answer deduce the location of the marble ((1/8)(3) + (1/8)(3)). Totaling up the average number of questions yields

$$\begin{aligned} \sum_{\boldsymbol{y}} -p(\boldsymbol{y}) \log_2 p(\boldsymbol{y}) &= \frac{1}{2}\log_2\frac{1}{2} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{8}\log_2\frac{1}{8} + \frac{1}{8}\log_2\frac{1}{8} \\ &= \frac{1}{2}(1) + \frac{1}{4}(2) + \frac{1}{8}(3) + \frac{1}{8}(3) = 1.75. \end{aligned}$$

It must be emphasized that the logarithms' *arguments* are consequences of *your* guessing strategy, but their *coefficients* are the result of *my* behavior across repeated plays of the game.

Now perhaps I think you're winning the game too often, so I try to push the average number of questions up by choosing a new hiding strategy:

$$q(\boldsymbol{Z}) = \begin{array}{|c|c|c|c|} \multicolumn{1}{c}{\text{A}} & \multicolumn{1}{c}{\text{B}} & \multicolumn{1}{c}{\text{C}} & \multicolumn{1}{c}{\text{D}} \\ \hline 1/4 & 1/4 & 1/4 & 1/4 \\ \hline \end{array}. \tag{4.2}$$

Intuitively, by making the "odds more even," I have made the problem harder; and indeed, even though you tailor your guessing strategy to these new odds, you must on average ask more questions to locate the marble:

$$\sum_{\boldsymbol{z}} -q(\boldsymbol{z}) \log_2 q(\boldsymbol{z}) = \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} = 2.0.$$

Information entropy. It should be intuitive that, by hiding the marble "more randomly," I have increased the number of yes/no questions you must ask to locate it. In a rather different context, Shannon famously sought a mathematical expression for the "randomness" or "uncertainty" of a random variable, $\boldsymbol{Y}$, in terms of its probability distribution, that captured two essential notions: that broader distributions should be more uncertain, and that the uncertainty of two independent random variables should be additive (as opposed to sub- or super-additive) [**?**, 12]. Remarkably, he showed that this expression must be (equivalent to) the number of yes/no questions asked in a guessing game like ours. More precisely, he showed that any expression satisfying his desiderata must be, up to a scale factor,

$$\sum_{\boldsymbol{y}} -p(\boldsymbol{y}) \log p(\boldsymbol{y}) =: \mathrm{H}_p[\boldsymbol{Y}], \tag{4.3}$$

a quantity he called *entropy* for its resemblance to the quantity of that name in statistical physics, and for which accordingly we use the symbol H (i.e., capital "eta" for *e*ntropy). Because the scaling is irrelevant, a logarithm of any base will do equally well. The natural logarithm is most mathematically convenient so we default to it throughout; but base 2 yields the convenient intepretation of our guessing game, and allows entropy to be denominated in the familiar quantity of bits.

information entropy

We return to that game now but suppose this time that you do *not* know my hiding probabilities. In particular, suppose I hide the marble according to the first scheme, Eq. 4.1, but you guess according to the second scheme, Eq. 4.2. Then the number of questions you will need to ask, on average, is

$$\sum_{\boldsymbol{y}} -p(\boldsymbol{y}) \log_2 q(\boldsymbol{y}) = \frac{1}{2}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{8}\log_2\frac{1}{4} + \frac{1}{8}\log_2\frac{1}{4} = 2.0.$$

It is no coincidence that more guesses will be required on average under the "wrong" distribution (2.0) than under the right one (1.75). Lest one suspect that this has to do with the entropies of p and q, notice that the result still holds when the distributions are reversed: If I hide the marble according to the uniform distribution (q, Eq. 4.2) and you guess according to the non-uniform distribution (p, Eq. 4.1), then you will likewise have to ask more questions than if you had guessed according to my distribution:

$$\sum_{\boldsymbol{z}} -q(\boldsymbol{z}) \log_2 p(\boldsymbol{z}) = \frac{1}{4}\log_2\frac{1}{2} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{8} + \frac{1}{4}\log_2\frac{1}{8} = 2.25,$$

rather than 2.0.

In general, we call the average number of questions required under the strategy derived from q, when the marble is hidden according to p, the *cross entropy* between p and q:

cross entropy

$$\sum_{\boldsymbol{y}} -p(\boldsymbol{y}) \log q(\boldsymbol{y}) =: \mathrm{H}_{pq}[\boldsymbol{Y}]. \tag{4.4}$$

That the cross entropy $\mathrm{H}_{pq}[\boldsymbol{Y}]$ is always greater than the entropy (*Gibbs's inequality*) follows

Gibbs's inequality

from Jensen's inequality (see Section **??**):

$$\begin{aligned}
\mathrm{H}_{pq}[\boldsymbol{Y}] - \mathrm{H}_{p}[\boldsymbol{Y}] &= \sum_{\boldsymbol{y}} -p(\boldsymbol{y}) \log q(\boldsymbol{y}) + \sum_{\boldsymbol{y}} p(\boldsymbol{y}) \log p(\boldsymbol{y}) && \text{Eqs. 4.3 and 4.4} \\
&= \mathbb{E}_{\boldsymbol{y}}\left[-\log \frac{q(\boldsymbol{y})}{p(\boldsymbol{y})}\right] \\
&\geq -\log\left(\mathbb{E}_{\boldsymbol{y}}\left[\frac{q(\boldsymbol{y})}{p(\boldsymbol{y})}\right]\right) && \text{Jensen's} \\
&= -\log(1) \\
&= 0,
\end{aligned}$$

with equality only when $p = q$ (again by Jensen's inequality). Since $q(\boldsymbol{Z})$ could be any distribution, this also shows that the guessing strategy proposed at the outset is the optimal one: no other strategy can yield fewer questions on average. The quantity on the left-hand side is, then, the number of *extra* questions you have to ask in virtue of having used the wrong strategy. This quantity also has a name, the *relative entropy*, or Kullback-Leibler (KL) divergence: *relative entropy*

$$\mathrm{D}_{\mathrm{KL}}\{p(\boldsymbol{Y}) || q(\boldsymbol{Y})\} := \mathrm{H}_{pq}[\boldsymbol{Y}] - \mathrm{H}_{p}[\boldsymbol{Y}]. \tag{4.5}$$

Efficient coding. The guessing-game interpretation of entropy maps straightfowardly onto the classical coding problem. Under an efficient code, more frequent data $\boldsymbol{y}$ are assigned shorter code words. And indeed, assigning 1 and 0 (resp.) to the answers "yes" and "no" in our guessing game yields the following binary codes for the four letters:

$$\mathrm{A} = 1 \qquad \mathrm{B} = 01 \qquad \mathrm{C} = 001 \qquad \mathrm{D} = 000.$$

This is a prefix-free code—no codeword has another codeword as a prefix—so any string of binary numbers, like 10011000101, has an unambiguous interpretation (ACADAB). The average codeword length is the entropy of the data—e.g., 1.75 bits under Eq. 4.1. As with the guessing game, designing the code (guessing strategy) according to the wrong distribution costs an extra number of bits (questions) given by the relative entropy of the correct to the incorrect distributions, Eq. 4.5. For example, Eq. 4.2 suggests partitioning the space first into (A or B) vs. (C or D), and thence into individual letters, which is equivalent to the coding scheme

$$\mathrm{A} = 11 \qquad \mathrm{B} = 10 \qquad \mathrm{C} = 01 \qquad \mathrm{D} = 00.$$

This is also a prefix-free code, but as we have seen, under Eq. 4.2 it costs 2.0 bits on average.

Thus the relative entropy measures how much less efficient it is to use q rather than p to encode some data $\boldsymbol{y}$ drawn from $p(\boldsymbol{Y})$. Since the relative entropy is non-negative and zero only when $p = q$, we have at last operationalized the notion of one distribution being or becoming "more like" another: We make a distribution q more like a distribution p when we decrease the number of extra bits required to encode, according to q, data drawn from $p(\boldsymbol{Y})$.

Minimizing relative entropy. Most (but not all) of the losses in this book, then, whether for discriminative or generative models, supervised or unsupervised learning, can

be written as relative entropies, with the model distribution $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$ in place of q. The generic optimization problem is

$$\begin{aligned}
\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\{\mathrm{D}_{\mathrm{KL}}\{p(\boldsymbol{Y})||\hat{p}(\boldsymbol{Y};\boldsymbol{\theta})\}\} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\{\mathrm{H}_{p\hat{p}}[\boldsymbol{Y};\boldsymbol{\theta}] - \mathrm{H}_{p}[\boldsymbol{Y}]\} && \text{by def'n, Eq. 4.5} \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\{\mathrm{H}_{p\hat{p}}[\boldsymbol{Y};\boldsymbol{\theta}]\} && \text{entropy is parameter-free} \\
&\approx \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\{\langle -\log \hat{p}(\boldsymbol{Y};\boldsymbol{\theta})\rangle_{\boldsymbol{Y}}\} && \text{approx. w/sample average} \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\{\langle \log \hat{p}(\boldsymbol{Y};\boldsymbol{\theta})\rangle_{\boldsymbol{Y}}\} \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left\{\log \prod_{n}^{N} \hat{p}(\boldsymbol{y}_n;\boldsymbol{\theta})\right\} \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left\{\prod_{n}^{N} \hat{p}(\boldsymbol{y}_n;\boldsymbol{\theta})\right\} && \log \text{ is monotonic.}
\end{aligned} \tag{4.6}$$

Thus we see that minimizing relative entropy is equivalent to minimizing cross entropy, or again to *maximizing the likelihood of the parameters*, thus connecting our optimization to the classical objective for fitting statistical models.

maximum likelihood

Maximum-likelihood estimates (MLEs) have long enjoyed widespread use in statistics for their asymptotic properties: Suppose the data were "actually" generated from a parameterized distribution within the family of model distributions, $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$. Then as the sample size approaches infinity, the MLE converges (in probability) to the true underlying parameter ("consistency") and achieves the mininum mean squared error among all consistent estimators ("efficiency"). The parameter estimates of models fit by minimizing relative entropy inherit these properties.

[.....]

[[Forward and reverse KL]]

[[continuous RVs]]

[[well known losses like squared error and the "binary cross entropy"]]

Chapter 5

Learning in Discriminative Models

The essential feature of discriminative models is that they do not attempt to model the distribution of all of the available data. Instead, they attempt to model only the distribution of one set of data conditioned on another set, $\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$. No attempt is made to model the distribution of $\hat{\boldsymbol{Y}}$. Consequently, the variables $\hat{\boldsymbol{Y}}$ and $\hat{\boldsymbol{X}}$ are often referred to as the "inputs" and "outputs," respectively.[1]

Since we are focused on parameteric models, the critical questions are:

1. What parametric family of distributions shall we use for the conditional? and
2. What family of functions shall we use for the map from the "inputs," $\hat{\boldsymbol{X}}$, to the parameters of that distribution?

5.1 Supervised Learning

In the classical notion of a discriminative model, the inputs and outputs are completely observed. That is, classically, learning in discriminative models is supervised. We shall nevertheless have occasion to explore unsupervised learning in discriminative models—in the next section. Here, we describe supervised learning as an instance of the general approach laid out in Section 4.2, to wit, minimizing relative or cross entropy. Since are only modeling the conditional distribution, this means the conditional entropies. But remember that an expectation is still taken under $p(\boldsymbol{Y})$:

$$\begin{aligned}\boldsymbol{\theta}^* &= \operatorname*{argmin}_{\boldsymbol{\theta}}\Big\{\mathrm{D}_{\mathrm{KL}}\Big\{p(\boldsymbol{X}\,|\,\boldsymbol{Y})\Big\|\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)\Big\}\Big\}\\ &= \operatorname*{argmin}_{\boldsymbol{\theta}}\{\mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}[-\log\hat{p}(\boldsymbol{X}\,|\,\boldsymbol{Y};\boldsymbol{\theta})]\}.\end{aligned} \tag{5.1}$$

How we proceed from here depends on our answers to the two questions posed at the outset.

[1] With some reservations, I have reversed the standard convention of using $\hat{\boldsymbol{Y}}$ for outputs and $\hat{\boldsymbol{X}}$ for inputs. The point is to emphasize that discriminative models are the Bayesian inverses of generative models. But why should the generative models use $\hat{\boldsymbol{X}}$ for their "source" variables and $\hat{\boldsymbol{Y}}$ for the emissions? This in turn is to match the standard conventions from control theory for, e.g., linear dynamical system; see for example Section 2.2. This tension is clearly felt in the machine-learning literature, where generative models typically introduce yet another symbol, $\hat{\boldsymbol{Z}}$, for their latent variables!

5.1.1 Linear regression

Here we begin our account of supervised learning with the model that gives the simplest answers: the (multivariate) normal distribution, whose mean depends linearly on the inputs:

$$\hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) = \tau^{-K/2}\Big|\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\Big|^{-1/2}\exp\Big\{-\frac{1}{2}\Big(\hat{\boldsymbol{X}}-\mathbf{G}\,\hat{\boldsymbol{Y}}\Big)^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}^{-1}\Big(\hat{\boldsymbol{X}}-\mathbf{G}\,\hat{\boldsymbol{Y}}\Big)\Big\}.$$

Note that this could easily be extended to an affine function, $\boldsymbol{\mu}_{\hat{x}|\hat{y}} = \mathbf{G}\,\hat{\boldsymbol{Y}} + \boldsymbol{g}$, but the notation is simpler if we assume that both the inputs and outputs, $\boldsymbol{X}$ and $\boldsymbol{Y}$, have been centered, in which case $\boldsymbol{g}$ is otiose. Alternatively, a fixed value of 1 can be appended to the vector $\hat{\boldsymbol{Y}}$, and $\boldsymbol{g}$ appended as a final column on $\mathbf{G}$—after all, we will make no assumptions about the distribution of $\hat{\boldsymbol{Y}}$.

To find $\mathbf{G}$, we differentiate the loss in Eq. 5.1:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\mathbf{G}}\mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}[-\log\hat{p}(\boldsymbol{X}\,|\,\boldsymbol{Y};\boldsymbol{\theta})] &= \mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}\left[\frac{\mathrm{d}}{\mathrm{d}\mathbf{G}}\frac{1}{2}(\boldsymbol{X}-\mathbf{G}\,\boldsymbol{Y})^{\mathrm{T}}\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}^{-1}(\boldsymbol{X}-\mathbf{G}\,\boldsymbol{Y})\right]\\ &= -\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}^{-1}\mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}\big[(\boldsymbol{X}-\mathbf{G}\,\boldsymbol{Y})\,\boldsymbol{Y}^{\mathrm{T}}\big] \overset{\text{set}}{=} 0\\ \implies \mathbf{G} &= \mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}\big[\boldsymbol{X}\,\boldsymbol{Y}^{\mathrm{T}}\big]\mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}\big[\boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\big]^{-1}\\ &\approx \big\langle\boldsymbol{X}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{X},\boldsymbol{Y}}\big\langle\boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}}^{-1},\end{aligned} \tag{5.2}$$

where the move to a sample average in the final line reflects the fact that we have only samples from the data distribution. The final line is famous enough to have earned its own name, the *normal equations*. Acquiring $\mathbf{G}$ through the normal equations is known as *linear regression*. In fact, our optimization is not the only route to the normal equations, and to gain more intuition about linear regression we shall examine several of these. But first we investigate a variation on the optimization just derived.

normal equations

linear regression

Regularization. The inverted term $\big\langle\boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}}$ is a sum of outer products. Therefore the resulting matrix, although obviously square, is not invertible if there are fewer samples than dimensions of $\boldsymbol{Y}$, in which case a unique solution for $\mathbf{G}$ does not exist. In that case a pseudo-inverse can be used to solve Eq. 5.2. However, as we shall see, the standard pseudo-inverse is merely a special case of a more general and elegant solution to the problem of *underdetermined* normal equations.

underdetermined

For notational simplicity, consider the special case of a scalar output—for example $\hat{X}_k$, entry k in the vector of outputs $\hat{\boldsymbol{X}}$. And now let us treat the corresponding row of $\mathbf{G}$ as a *random* vector, $\boldsymbol{G}_k^{\mathrm{T}}$, independent of $\hat{\boldsymbol{Y}}$ and with its own (generic) prior distribution:[2]

$$\hat{p}\Big(\boldsymbol{G}_k\,\Big|\,\hat{\boldsymbol{Y}}\Big) = \hat{p}(\boldsymbol{G}_k) = \frac{1}{Z_g}\exp\big\{-E_k(\boldsymbol{G}_k)\big\}. \tag{5.3}$$

Then in place of original conditional distribution, $\hat{p}\Big(X_k\,\Big|\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$, we use the augmented conditional $\hat{p}\Big(X_k,\boldsymbol{G}_k\,\Big|\,\hat{\boldsymbol{Y}}\,k\Big)$, a kind of posterior distribution over $\boldsymbol{G}_k$:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{g}_k{}^{\mathrm{T}}}\mathbb{E}_{X_k,\boldsymbol{Y}}\Big[-\log\hat{p}\Big(X_k,\boldsymbol{g}_k\,\Big|\,\boldsymbol{Y}\Big)\Big] &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{g}_k{}^{\mathrm{T}}}\mathbb{E}_{X_k,\boldsymbol{Y}}\Big[-\log\Big(\hat{p}\Big(X_k\,\Big|\,\boldsymbol{Y},\boldsymbol{g}_k\Big)\hat{p}\Big(\boldsymbol{g}_k\Big)\Big)\Big]\\ &= -\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}^{-1}\mathbb{E}_{X_k,\boldsymbol{Y}}\Big[\Big(X_k-\boldsymbol{g}_k\,\boldsymbol{Y}\Big)\,\boldsymbol{Y}^{\mathrm{T}}\Big] + \frac{\mathrm{d}E_k}{\mathrm{d}\boldsymbol{g}_k{}^{\mathrm{T}}}(\boldsymbol{g}_k) \overset{\text{set}}{=} 0\end{aligned}$$

[2]Recall that random vectors use bold italic capitals, whereas matrices are assigned bold Roman capitals.

For example, consider an isotropic, zero-mean, Gaussian prior distribution over $\boldsymbol{G}_k$. Intuitively, this encodes our belief that the parameters are *most likely to be zero*, with the penalty for being non-zero growing quadratically with distance. No parameter is considered *a priori* any more important than, or correlated with, any other. Furthermore, the energy and its gradient for the isotropic, zero-mean, Gaussian are

$$E_k(\boldsymbol{g}_k) = \frac{1}{2}\boldsymbol{g}_k{}^{\mathrm{T}}\boldsymbol{g}_k, \qquad \frac{\mathrm{d}E_k}{\mathrm{d}\boldsymbol{g}_k{}^{\mathrm{T}}}(\boldsymbol{g}_k) = \boldsymbol{g}_k{}^{\mathrm{T}}, \tag{5.4}$$

so we can solve for $\mathbf{G}$ in closed-form:

$$\begin{aligned} 0 &= -\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}^{-1}\mathbb{E}_{X_k,\boldsymbol{Y}}\Big[\Big(X_k - \boldsymbol{g}_k\,\boldsymbol{Y}\Big)\,\boldsymbol{Y}^{\mathrm{T}}\Big] + \boldsymbol{g}_k{}^{\mathrm{T}} \\ \implies 0 &= -\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}^{-1}\mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}\big[(\boldsymbol{X} - \mathbf{G}\,\boldsymbol{Y})\,\boldsymbol{Y}^{\mathrm{T}}\big] + \mathbf{G} \\ \implies \mathbf{G} &= \big\langle \boldsymbol{X}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{X},\boldsymbol{Y}}\Big(\big\langle \boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}} + \boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\Big)^{-1}. \end{aligned} \tag{5.5}$$

[[Since the energy in Eq. 5.4 represents an L^2 norm, this is known as *Tikhonov, ridge, or L^2-regularized regression*.]] As long as the rank of $\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}$ is full, so is the rank of the inverted term (adding positive definite matrices cannot reduce rank), so the normal equations are now solvable even for rank-deficient $\big\langle \boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}}$.

Tikhonov, ridge, or L^2-regularized regression

Newton-Raphson. The least-squares penalty can be solved in one step because the resulting cost function is quadratic in $\mathbf{G}$. Still, for costs that are not quadratic, but nevertheless convex, the (celebrated) Newton-Raphson algorithm is guaranteed to find the unique global solution. In anticipation of encountering such costs (Section 5.1.2), we apply this method to linear regression.

The basic Newton-Raphson method aims to find the roots (zeros) of a scalar function of a scalar input, $g(\theta)$, as follows. From a given point in parameter space, $\theta^{(i)}$, move along the local line tangent to $g(\theta^{(i)})$ to the point where it intercepts the θ axis; call this point $\theta^{(i+1)}$ and iterate. For appropriately smooth functions, each root θ^*, at which $g(\theta^*) = 0$, is guaranteed to be the convergent value of this procedure when intialized from some surrounding neighborhood. Mathematically, the algorithm amounts to enforcing

$$\text{slope} = \frac{\text{rise}}{\text{run}} \implies g'(\theta^{(i)}) \stackrel{\text{set}}{=} \frac{g(\theta^{(i)}) - 0}{\theta^{(i)} - \theta^{(i+1)}} \implies \theta^{(i+1)} = \theta^{(i)} - g(\theta^{(i)})/g'(\theta^{(i)}).$$

The procedure is easily extended to vector-valued functions of vector inputs, as long as the vectors have the same length (otherwise the extension is more complicated). This is especially germane to our problem of finding extrema of a function $f(\boldsymbol{\theta})$, since the roots of the function $\boldsymbol{g} := f'$ are extrema of f. Replacing the scalar-valued functions above with their vector and matrix counterparts yields:

$$\boldsymbol{\theta}^{(i+1)^{\mathrm{T}}} = \boldsymbol{\theta}^{(i)^{\mathrm{T}}} - \frac{\partial f}{\partial \boldsymbol{\theta}^{\mathrm{T}}}(\boldsymbol{\theta}^{(i)})\left(\frac{\partial^2 f}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}^{\mathrm{T}}}(\boldsymbol{\theta}^{(i)})\right)^{-1}. \tag{5.6}$$

In our case, the parameters are in the form of a matrix, $\mathbf{G}$, which would make the Hessian a tensor. To avoid this ugliness, let us work with one row of the matrix, $\boldsymbol{g}_k{}^{\mathrm{T}}$, at a time. Then from Eq. 5.2, we see that the gradient and Hessian of the cross entropy H are

$$\frac{\partial H}{\partial \boldsymbol{g}_k}(\boldsymbol{g}_k) = -\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}^{-1}\mathbb{E}_{X,\boldsymbol{Y}}\big[\big(X_k - \boldsymbol{g}_k{}^{\mathrm{T}}\,\boldsymbol{Y}\big)\,\boldsymbol{Y}\big], \qquad \frac{\partial^2 H}{\partial \boldsymbol{g}_k \partial \boldsymbol{g}_k{}^{\mathrm{T}}}(\boldsymbol{g}_k) = \boldsymbol{\Sigma}_{\hat{x}|\hat{y}}^{-1}\mathbb{E}_{\boldsymbol{Y}}\big[\boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\big].$$

Consequently, the Newton-Raphson update becomes

$$\begin{aligned}
{\boldsymbol{g}_k}^{(i+1)^{\mathrm{T}}} &= {\boldsymbol{g}_k}^{(i)^{\mathrm{T}}} + \mathbb{E}_{X,\boldsymbol{Y}}\Big[\Big(X_k - {\boldsymbol{g}_k}^{(i)^{\mathrm{T}}}\boldsymbol{Y}\Big)\boldsymbol{Y}^{\mathrm{T}}\Big]\mathbb{E}_{\boldsymbol{Y}}\big[\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}\big]^{-1} \\
&= {\boldsymbol{g}_k}^{(i)^{\mathrm{T}}} + \Big(\mathbb{E}_{X,\boldsymbol{Y}}\big[X_k\,\boldsymbol{Y}^{\mathrm{T}}\big] - {\boldsymbol{g}_k}^{(i)}\mathbb{E}_{\boldsymbol{Y}}\big[\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}\big]\Big)\mathbb{E}_{\boldsymbol{Y}}\big[\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}\big]^{-1} \\
&= \mathbb{E}_{X,\boldsymbol{Y}}\big[X_k\,\boldsymbol{Y}^{\mathrm{T}}\big]\mathbb{E}_{\boldsymbol{Y}}\big[\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}\big]^{-1} \\
\implies \mathbf{G} &= \mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}\big[\boldsymbol{X}\boldsymbol{Y}^{\mathrm{T}}\big]\mathbb{E}_{\boldsymbol{Y}}\big[\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}\big]^{-1},
\end{aligned}$$

where on the last line we have simply collected up the updates for all rows. We see that for any starting point, the solution (which doesn't depend on that point) is reached in one step, as expected.

Moment matching under additive noise. Linear regression is so ubiquitous that the assumption of Gaussian noise, although frequently justifiable by reference to the central limit theorem, may feel overly restrictive. Let us therefore retain the assumption of a linear map with additive, independent, zero-mean noise,

$$\hat{\boldsymbol{X}} = \mathbf{G}\,\hat{\boldsymbol{Y}} + \hat{\boldsymbol{Z}}, \tag{5.7}$$

but no longer require the noise, $\hat{\boldsymbol{Z}}$, to be normally distributed. Now, without a probability model, it is no longer possible to minimize the relative entropy between data and model distributions. Instead, we will match their first two moments. More precisely, we shall require that

$$\begin{aligned}
\mathbb{E}_{\hat{\boldsymbol{X}}}\Big[\hat{\boldsymbol{X}}\Big] &\overset{\text{set}}{=} \langle \boldsymbol{X}\rangle_{\boldsymbol{X}} & \mathbb{E}_{\hat{\boldsymbol{Y}}}\Big[\hat{\boldsymbol{Y}}\Big] &\overset{\text{set}}{=} \langle \boldsymbol{Y}\rangle_{\boldsymbol{Y}} \\
\mathbb{E}_{\hat{\boldsymbol{X}},\hat{\boldsymbol{Y}}}\Big[\hat{\boldsymbol{X}}\,\hat{\boldsymbol{Y}}^{\mathrm{T}}\Big] &\overset{\text{set}}{=} \big\langle \boldsymbol{X}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{X},\boldsymbol{Y}} & \mathbb{E}_{\hat{\boldsymbol{Y}}}\Big[\hat{\boldsymbol{Y}}\,\hat{\boldsymbol{Y}}^{\mathrm{T}}\Big] &\overset{\text{set}}{=} \big\langle \boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}}.
\end{aligned}$$

Notice that we do not need to specify the expected outer product of the outputs. Using the fact that the noise is independent of the inputs, we see that

$$\mathbb{E}_{\hat{\boldsymbol{X}}}\Big[\hat{\boldsymbol{X}}\,\hat{\boldsymbol{X}}^{\mathrm{T}}\Big] = \mathbb{E}_{\hat{\boldsymbol{Y}},\hat{\boldsymbol{Z}}}\Big[(\mathbf{G}\,\hat{\boldsymbol{Y}} + \hat{\boldsymbol{Z}})(\mathbf{G}\,\hat{\boldsymbol{Y}} + \hat{\boldsymbol{Z}})^{\mathrm{T}}\Big] = \mathbf{G}\mathbb{E}_{\hat{\boldsymbol{Y}}}\Big[\hat{\boldsymbol{Y}}\,\hat{\boldsymbol{Y}}^{\mathrm{T}}\Big]\mathbf{G}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\hat{z}}.$$

So assuming $\big\langle \boldsymbol{X}\,\boldsymbol{X}^{\mathrm{T}}\big\rangle_{\boldsymbol{X}}$ is "bigger" than $\mathbf{G}\big\langle \boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}}\mathbf{G}^{\mathrm{T}}$, the noise covariance $\boldsymbol{\Sigma}_{\hat{z}}$ can always make up the difference.

Furthermore, as lately noted, equality of means can be achieved simply by centering the data. That leaves $\big\langle \boldsymbol{X}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{X},\boldsymbol{Y}}$ and $\big\langle \boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}}$. Applying our "model" (Eq. 5.7), using the fact that the means are zero, and then setting the model expectations equal to the data expectations, yields:

$$\begin{aligned}
\mathbb{E}_{\hat{\boldsymbol{X}},\hat{\boldsymbol{Y}}}\Big[\hat{\boldsymbol{X}}\,\hat{\boldsymbol{Y}}^{\mathrm{T}}\Big] = \mathbb{E}_{\hat{\boldsymbol{Y}}}\Big[(\mathbf{G}\,\hat{\boldsymbol{Y}} + \hat{\boldsymbol{Z}})\,\hat{\boldsymbol{Y}}^{\mathrm{T}}\Big] &= \mathbf{G}\mathbb{E}_{\hat{\boldsymbol{Y}}}\Big[\hat{\boldsymbol{Y}}\,\hat{\boldsymbol{Y}}^{\mathrm{T}}\Big] \\
\implies \mathbf{G} &\approx \big\langle \boldsymbol{X}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{X},\boldsymbol{Y}}\big\langle \boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}}^{-1},
\end{aligned}$$

the normal equations.

In fine, assuming that the additive noise is Gaussian has the same net effect as fitting a kind of second-order approximation to the joint distribution. This should not be surprising, since the normal distribution is the most "agnostic" of all distributions that specify the first two moments.

The emission covariance. Let us pause briefly to consider the meaning of, and whether we should be surprised by, the fact that $\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}$ disappears from Eq. 5.2. It implies that use of the normal equations makes no assumptions about the "shape" (covariance) of the noise about points in the output space. For example, some outputs could be "more important" (lower-variance) than, or correlated with, others. These correspond respectively to differing values on the diagonal, and non-zero values in the off-diagonals, of the covariance matrix. The reason this makes no difference to the solution is that we have at our disposal a separate row in $\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}$ for every output, $\hat{X}_k$: we can safely adjust weights for one output dimension without affecting any other.

But not under regularization; explain this.

What we do not have is a separate set of parameters for every *pair of samples*, $\boldsymbol{x}_n, \boldsymbol{y}_n$. This caused no complications because the samples were assumed to be i.i.d. Nevertheless, the assumption of i.i.d. samples is not always appropriate. For example.... The data are then *heteroscedastic* (differently dispersed) rather than *homoscedastic* (identically dispersed). Still, if the dependence between samples is only second-order—i.e., in the first-order correlations—and known, the linear regression has an only slightly more complicated complete solution. It is most most elegantly derived when the heteroscedastic data samples are represented explicitly.

heteroscedastic
homoscedastic

Linear regression without probability distributions

Accordingly, let us assemble all the samples into two matrices:[3]

$$\mathbf{X} := \begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_N \end{bmatrix} \qquad\qquad \mathbf{Y} := \begin{bmatrix} \boldsymbol{y}_1 & \cdots & \boldsymbol{y}_N \end{bmatrix}.$$

In place of the preceding models, let us simply look for a solution to the linear equation

$$\mathbf{X} \overset{\text{set}}{=} \mathbf{G}\mathbf{Y}. \tag{5.8}$$

The equation has a unique solution if and only if $\mathbf{Y}$ is square—that is, there are precisely as many samples (N) as dimensions (M)—and full-rank. When $\mathbf{Y}$ is "tall" (more dimensions than samples, $M > N$), the solution is underdetermined. We have postponed this variant of the problem and will continue to do so for now. When $\mathbf{Y}$ is "fat" (more samples than dimensions, $N > M$), the solution is overdetermined: *no* matrix $\mathbf{G}$ satisfies Eq. 5.8, and a notion of "best fit" must be imposed in order to select just one of the many approximately satisfactory matrices.

Still, let us proceed somewhat naïvely and simply look for an obvious linear-algebraic solution. If $\mathbf{Y}$ is full-rank in addition to being tall, then the Gram matrix (in sample space) $\mathbf{Y}\mathbf{Y}^{\mathrm{T}}$ is square ($M \times M$) and full-rank, and therefore invertible. Hence, starting from Eq. 5.8,

$$\mathbf{X}\mathbf{Y}^{\mathrm{T}} = \mathbf{G}\mathbf{Y}\mathbf{Y}^{\mathrm{T}} \implies \mathbf{X}\mathbf{Y}^{\mathrm{T}}\big(\mathbf{Y}\mathbf{Y}^{\mathrm{T}}\big)^{-1} = \mathbf{G}.$$

Again we have recovered the normal equations, this time apparently with even fewer assumptions. Lest the normal equations seem inevitable, let us recall that the choice to right multiply by the matrix $\mathbf{Y}^{\mathrm{T}}$ was, although perhaps obvious, also arbitrary. Note in particular that, under our assumption that $\mathbf{Y}$ is tall, choosing $\mathbf{G}$ according to the normal equations *does not actually satisfy Eq. 5.8*—indeed, no choice of $\mathbf{G}$ could, because the problem is

[3] These are the transposes of the matrices used in most developments of linear regression, but they are consistent with all the other standard conventions used throughout this book.

overdetermined. What notion of "best fit" have we tacitly imposed in order to select one value for $\mathbf{G}$?

The answer can be read off the derivation in Eq. 5.2 above: The normal equations arise from *squared-error* penalties, and accordingly the procedure is sometimes called the *method of least squares*. Evidently the assumption of normally distributed errors and the least-squares penalty are two sides of the same coin. In the case of a linear map, when $\mathbf{Y}$ is full rank, the least-squares solution exists. And since the quadratic is convex, the solution is the unique; that is, the normal equations yield the global minimizer of the least-squares penalty. As a "sanity check," we can also reassure ourselve that when Eq. 5.8 really does hold, i.e. when $\mathbf{Y}$ is invertible, the normal equations reduce to the unique solution, $\mathbf{G} = \mathbf{X}\mathbf{Y}^{-1}$.

method of least squares

Heteroscedastic data: weighted least-squares. Now that we have examined linear regression thoroughly from the perspective of the data matrices $\mathbf{X}$ and $\mathbf{Y}$, let us complicate the problem along the lines suggested above. Suppose that the samples in these matrices are not i.i.d. In particular, let $\tilde{\boldsymbol{X}}_k$ and $\tilde{\boldsymbol{Y}}_m$ be random vectors of all samples, for dimensions k and m of the output and input (resp.), and suppose $\mathrm{Cov}_{\tilde{\boldsymbol{X}}_k \mid \tilde{\boldsymbol{Y}}_m}\left[\tilde{\boldsymbol{X}}_k \,\middle|\, \tilde{\boldsymbol{Y}}_m\right] = \boldsymbol{\Upsilon}$ (fixed for all k and m). Then we can decorrelate $\tilde{\boldsymbol{X}}_k$ simply by multiplying by $\boldsymbol{\Upsilon}^{-1/2}$. In terms of the data matrix $\mathbf{X}$, in which each $\tilde{\boldsymbol{x}}_k$ is a *row*, not a column, this amounts to right multiplication. To maintain the linear relationship in Eq. 5.8, we multiply *both* sides by $\boldsymbol{\Upsilon}^{-1/2}$. Replacing $\mathbf{X}$ with $\mathbf{X}\boldsymbol{\Upsilon}^{-1/2}$ and $\mathbf{Y}$ with $\mathbf{Y}\boldsymbol{\Upsilon}^{-1/2}$ turns the normal equations into

$$\mathbf{G} = \mathbf{X}\boldsymbol{\Upsilon}^{-1}\mathbf{Y}^{\mathrm{T}}(\mathbf{Y}\boldsymbol{\Upsilon}^{-1}\mathbf{Y}^{\mathrm{T}})^{-1}. \tag{5.9}$$

This variation is known as *weighted least squares*.

weighted least squares

Geometric arguments. [[XXX]]

5.1.2 Generalized linear models

Having explored rather thoroughly the simplest model, let us revisit the methodological choices posed at the beginning of the chapter. In particular, suppose we relax the assumption that the conditional be Gaussian, but retain the assumption that the data are combined linearly across dimensions. [[Some discussion of finite sufficient statistics to motivate exponential families....]] This class includes the multivariate Gaussian, but also many other familiar distirbutions—Poisson, multinomial, Gamma, Dirichlet, etc.—as special cases. On the one hand, for none of these other distribution is the cross entropy quadratic in the parameters (recall Eq. 5.2), so closed-form minimizations are not typically possible. On the other hand, the cross-entropy loss does retain convexity for any exponential-family distribution, so the Newton-Raphson algorithm lately derived is guaranteed to find the global optimum. As we shall see, each step of the algorithm can be rewritten as solving a *weighted* least-squares problem, as in Eq. 5.9, except that the weights change at every iteration.

Exponential families

Marginal distribution:

$$\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\eta}\Big) = h(\hat{\boldsymbol{X}})\exp\Big\{\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{t}(\hat{\boldsymbol{X}}) - A(\boldsymbol{\eta})\Big\} \tag{5.10}$$

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\eta}^{\mathrm{T}}}\mathbb{E}_{\boldsymbol{X}}[-\log\hat{p}(\boldsymbol{X};\boldsymbol{\eta})] = -\mathbb{E}_{\boldsymbol{X}}\left[\boldsymbol{t}^{\mathrm{T}}(\hat{\boldsymbol{X}}) - \frac{\mathrm{d}A}{\mathrm{d}\boldsymbol{\eta}^{\mathrm{T}}}(\boldsymbol{\eta})\right] \stackrel{\text{set}}{=} 0$$
$$\implies \mathbb{E}_{\boldsymbol{X}}[\boldsymbol{t}(\boldsymbol{X})] = \mathbb{E}_{\hat{\boldsymbol{X}}}\Big[\boldsymbol{t}(\hat{\boldsymbol{X}})\Big].$$

The optimal moment parameters occur when the gradient with respect to the natural parameters is zero....

Generalized linear model, assuming canonical response function:

$$\hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\hat{\boldsymbol{Y}};\mathbf{G}\Big) = h(\hat{\boldsymbol{X}})\exp\left\{\frac{\hat{\boldsymbol{Y}}^{\mathrm{T}}\mathbf{G}^{\mathrm{T}}\hat{\boldsymbol{X}} - A\Big(\mathbf{G}\,\hat{\boldsymbol{Y}}\Big)}{\phi}\right\} \tag{5.11}$$

Gradient:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\mathbf{G}}\mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}[-\log\hat{p}(\boldsymbol{X}\,|\,\boldsymbol{Y};\mathbf{G})] &= -\mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}\left[\frac{1}{\phi}\left(\boldsymbol{X} - \frac{\mathrm{d}A}{\mathrm{d}\boldsymbol{\eta}}\right)\boldsymbol{Y}^{\mathrm{T}}\right] \\ &= -\mathbb{E}_{\boldsymbol{Y}}\left[\frac{1}{\phi}\Big(\mathbb{E}_{\boldsymbol{X}\,|\,\boldsymbol{Y}}[\boldsymbol{X}\,|\,\boldsymbol{Y}] - \mathbb{E}_{\hat{\boldsymbol{X}}\,|\,\boldsymbol{Y}}\Big[\hat{\boldsymbol{X}}\,\Big|\,\boldsymbol{Y}\Big]\Big)\boldsymbol{Y}^{\mathrm{T}}\right]\end{aligned} \tag{5.12}$$

Hessian, for a scalar output:

$$\frac{\mathrm{d}^2}{\mathrm{d}\boldsymbol{g}\,\mathrm{d}\boldsymbol{g}^{\mathrm{T}}}\mathbb{E}_{X,\boldsymbol{Y}}[-\log\hat{p}(X\,|\,\boldsymbol{Y};\boldsymbol{g})] = \mathbb{E}_{\boldsymbol{Y}}\left[\frac{1}{\phi}\mathrm{Var}_{\hat{X}\,|\,\boldsymbol{Y}}\Big[\hat{X}\,\Big|\,\boldsymbol{Y}\Big]\,\boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\right] \tag{5.13}$$

IRLS update:

$$\begin{aligned}\boldsymbol{g}^{(i+1)^{\mathrm{T}}} &= \boldsymbol{g}^{(i)^{\mathrm{T}}} + \mathbb{E}_{\boldsymbol{Y}}\Big[\Big(\mathbb{E}_{X\,|\,\boldsymbol{Y}}[X\,|\,\boldsymbol{Y}] - \mathbb{E}_{\hat{X}\,|\,\boldsymbol{Y}}\Big[\hat{X}\,\Big|\,\boldsymbol{Y}\Big]\Big)\boldsymbol{Y}^{\mathrm{T}}\Big]\mathbb{E}_{\boldsymbol{Y}}\Big[\mathrm{Var}_{\hat{X}\,|\,\boldsymbol{Y}}\Big[\hat{X}\,\Big|\,\boldsymbol{Y}\Big]\,\boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\Big]^{-1} \\ &= \boldsymbol{g}^{(i)^{\mathrm{T}}} + \Big(\boldsymbol{x} - \boldsymbol{\mu}^{(i)}_{\hat{x}|\hat{y}}\Big)^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\big(\mathbf{Y}\mathbf{V}\mathbf{Y}^{\mathrm{T}}\big)^{-1} \\ &= \Big(\boldsymbol{g}^{(i)^{\mathrm{T}}}\mathbf{Y}\mathbf{V}\mathbf{Y}^{\mathrm{T}} + \Big(\boldsymbol{x} - \boldsymbol{\mu}^{(i)}_{\hat{x}|\hat{y}}\Big)^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\Big)\big(\mathbf{Y}\mathbf{V}\mathbf{Y}^{\mathrm{T}}\big)^{-1} \\ &= \Big(\boldsymbol{g}^{(i)^{\mathrm{T}}}\mathbf{Y} + \Big(\boldsymbol{x} - \boldsymbol{\mu}^{(i)}_{\hat{x}|\hat{y}}\Big)^{\mathrm{T}}\mathbf{V}^{-1}\Big)\mathbf{V}\mathbf{Y}^{\mathrm{T}}\big(\mathbf{Y}\mathbf{V}\mathbf{Y}^{\mathrm{T}}\big)^{-1} \\ &= \Big(\boldsymbol{\eta}^{(i)} + \Big(\boldsymbol{x} - \boldsymbol{\mu}^{(i)}_{\hat{x}|\hat{y}}\Big)^{\mathrm{T}}\mathbf{V}^{-1}\Big)\mathbf{V}\mathbf{Y}^{\mathrm{T}}\big(\mathbf{Y}\mathbf{V}\mathbf{Y}^{\mathrm{T}}\big)^{-1}.\end{aligned} \tag{5.14}$$

This is a weighted least-squares problem—cf. Eq. 5.9—but with the parenthetical quantity, rather than $\boldsymbol{x}$, as the output variable. Notice, however, that $\mathbf{V}^{-1} = \mathrm{d}\boldsymbol{\eta}^{(i)}/\,\mathrm{d}\boldsymbol{\mu}^{(i)}_{\hat{x}|\hat{y}}$. So the parenthetical quantity is the Taylor-series expansion of the natural parameters $\boldsymbol{\eta}^{(i)}$ in the vicinity of the mean, $\boldsymbol{\mu}^{(i)}_{\hat{x}|\hat{y}}$.

Iteratively reweighted least squares

IRLS for the Bernoulli distribution: Logistic regression

IRLS for the Poisson Distribution

Having derived IRLS, we get a feel for the algorithm by applying it to a few cases, beginnning with *Poisson* noise. For simplicity, we consider a single, scalar "output" or dependent

variable; and the *canonical* link function, the natural logarithm. This makes the mean of the distribution an exponentiated (inverse canonical link) linear function of the "input" or independent variables, $\hat{\boldsymbol{Y}}$, i.e. $\mathbb{E}_{\hat{X}|\boldsymbol{Y}}\left[\hat{X}\middle|\boldsymbol{Y}\right] = \exp\left\{\boldsymbol{g}^{\mathrm{T}}\hat{\boldsymbol{Y}}\right\}$. Recall that for Poisson random variables, the variance is equal to the mean. Inserting these values into the first line of Eq. 5.14, yields

$$\boldsymbol{g}^{(i+1)^{\mathrm{T}}} = \boldsymbol{g}^{(i)^{\mathrm{T}}} + \mathbb{E}_{X,\boldsymbol{Y}}\left[\left(X - \exp\left\{\boldsymbol{g}^{(i)^{\mathrm{T}}}\boldsymbol{Y}\right\}\right)\boldsymbol{Y}^{\mathrm{T}}\right]\mathbb{E}_{\boldsymbol{Y}}\left[\exp\left\{\boldsymbol{g}^{(i)^{\mathrm{T}}}\boldsymbol{Y}\right\}\boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\right]^{-1}.$$

IRLS for the Gamma Distribution

Update to new notation and perhaps rewrite.

We consider the case where the *scale* parameter is known, but the *shape* parameter is not. The reverse is much more commonly considered (it is the "gamma" GLiM built into MATLAB's `glmfit`, for example), probably because the sufficient statistic for the shape parameter is just the sum over independent output samples, $\sum_n \hat{x}_n$. For simplicity, we consider outputs of length one ($K = 1$) only, which makes the parameters a (row) vector, $\boldsymbol{\theta}^{\mathrm{T}}$. Now, the log-likelihood of the parameters under the gamma distribution is:

$$\begin{aligned}
\boldsymbol{\theta}^* &= \operatorname*{argmax}_{\boldsymbol{\theta}}\left\{\log\prod_n^N q(\hat{x}_n|\hat{\boldsymbol{y}}_n;\boldsymbol{\theta})\right\} \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}}\left\{\sum_n^N \log q(\hat{x}_n|\hat{\boldsymbol{y}}_n;\boldsymbol{\theta})\right\} \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}}\left\{\sum_n^N \left(\begin{bmatrix}\log\hat{x}_n\\ \hat{x}_n\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\eta_k^n(\hat{\boldsymbol{y}}_n,\boldsymbol{\theta})\\ \eta_\theta^n\end{bmatrix} - \log\Gamma(\eta_k^n(\hat{\boldsymbol{y}}_n,\boldsymbol{\theta})+1) + (\eta_k^n(\hat{\boldsymbol{y}}_n,\boldsymbol{\theta})+1)\log(-\eta_\theta^n)\right)\right\} \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}}\left\{\sum_n^N \left(\begin{bmatrix}\log\hat{x}_n\\ \hat{x}_n\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}k_n(\hat{\boldsymbol{y}}_n,\boldsymbol{\theta})-1\\ -1/\theta\end{bmatrix} - \log\Gamma(k_n(\hat{\boldsymbol{y}}_n,\boldsymbol{\theta})) - k_n(\hat{\boldsymbol{y}}_n,\boldsymbol{\theta})\log\theta\right)\right\}
\end{aligned}$$

Here we have made explicit that the only natural parameters depending on the inputs—and the parameters—are those associated with the shape parameter: $\eta_k^n = k_n - 1$. The scale parameter is assumed constant across all samples. Since k_n must be positive, we define the link function as:

$$\eta_k^n = \exp\left\{\boldsymbol{\theta}^{\mathrm{T}}\hat{\boldsymbol{y}}_n\right\} - 1.$$

The trailing 1 only clutters the derivation, so we work instead with k_n, starting with its gradient with respect to the parameters:

$$k_n = \exp\{\boldsymbol{\theta}^{\mathrm{T}}\hat{\boldsymbol{y}}_n\} \implies \frac{\mathrm{d}k_n}{\mathrm{d}\boldsymbol{\theta}^{\mathrm{T}}} = \exp\{\boldsymbol{\theta}^{\mathrm{T}}\hat{\boldsymbol{y}}_n\}\hat{\boldsymbol{y}}_n{}^{\mathrm{T}} = k_n\hat{\boldsymbol{y}}_n{}^{\mathrm{T}}.$$

We shall also need the first and second the derivatives of the log-partition function with respect to η_k^n, i.e., the expected value and variance of $\log\hat{x}_n$:

$$\begin{aligned}
\frac{\mathrm{d}A}{\mathrm{d}\eta_k^n} &= \frac{\mathrm{d}A}{\mathrm{d}k_n} = \psi^{(0)}(k_n) + \log\theta =: \mu_n, \\
\frac{\mathrm{d}\mu_n}{\mathrm{d}\eta_k^n} &= \frac{\mathrm{d}\mu_n}{\mathrm{d}k_n} = \psi^{(1)}(k_n) =: \sigma_n^2,
\end{aligned} \tag{5.15}$$

with $\psi^{(i)}(\cdot)$ the i^{th}-order polygamma function. Putting these pieces together, the gradient (with respect to the parameters) of the entire cost function above is

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{\theta}^{\mathrm{T}}} &= \sum_{n}^{N} (\log \hat{x}_n - \mu_n) k_n \hat{\boldsymbol{y}}_n{}^{\mathrm{T}} \\ &= \boldsymbol{v}\end{aligned}$$

where $v_n = (\log \hat{x}_n - \mu_n)k_n$.

The Newton-Raphson algorithm also requires the Hessian. Differentiating a second time we find

$$\begin{aligned}\frac{\partial^2 L}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^{\mathrm{T}}} &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}^{\mathrm{T}}} \sum_{n}^{N} \hat{\boldsymbol{y}}_n (\log \hat{x}_n - \mu_n) k_n \\ &= \sum_{n}^{N} \hat{\boldsymbol{y}}_n \Big(- \frac{\mathrm{d}\mu_n}{\mathrm{d}k_n{}^{\mathrm{T}}} \frac{\mathrm{d}k_n}{\mathrm{d}\boldsymbol{\theta}^{\mathrm{T}}} k_n + (\log \hat{x}_n - \mu_n) \frac{\mathrm{d}k_n}{\mathrm{d}\boldsymbol{\theta}^{\mathrm{T}}} \Big) \\ &= \sum_{n}^{N} \hat{\boldsymbol{y}}_n \big(- \sigma_n^2 k_n + \log \hat{x}_n - \mu_n \big) k_n \hat{\boldsymbol{y}}_n{}^{\mathrm{T}} \\ &= \mathbf{D},\end{aligned}$$

with $\mathbf{D}$ a diagonal matrix with entries $d_{nn} = -k_n^2\sigma_n^2 + k_n \log \hat{x}_n - k_n\mu_n$. Notice, however, that if we use the *expected* Hessian, the term $\log \hat{x}_n - \mu_n$ vanishes, leaving $d_{nn} = -k_n^2\sigma_n^2$.

The name "regression."

5.1.3 Artificial neural networks

Backpropagation

Update to new notation and begin with statistical formulation.

How do we train multi-layer neural networks to map certain inputs ($\hat{\boldsymbol{y}}$) into certain outputs ($\hat{\boldsymbol{x}}$)? This is the classical supervised-learning problem for neural networks; and the classical answer, still ubiquitous today, is "backprop." The procedure assigns blame to weights (synaptic strengths) in the network by propagating error derivatives backwards through the network, from the output to the input, and then changes the weights in proportion to their blameworthiness—an instance of gradient descent. The derivation is essentially just an application of the chain rule of differentiation from multivariate calculus to a rather complicated function, which necessitates some punctilious bookkeeping. In our discussion, we consider only feedforward neural nets with a single output layer at the end of the network (Fig. **??**); but more general variants are more or less easily derived from the equations presented here. The idea here is to introduce the concepts and (perhaps especially) the machinery for deriving the update rules. With a flexible matrix calculus in hand (see Section A.1), the derivations are straightforward, economical, and elegant.

Backprop is in fact the answer to a slightly more general question than that posed at the outset: How should the weights of an N-layer neural network be changed so as to reduce a loss, $L(\boldsymbol{u}_N, \hat{\boldsymbol{x}})$, evaluated on the output layer, $\boldsymbol{u}_N$? (Note that these outputs, $\boldsymbol{u}_N$, are deterministic functions of the network's inputs, $\hat{\boldsymbol{y}}$, so the loss is actually a function of both inputs and outputs). When the goal is to make the network produce certain output "targets," $\hat{\boldsymbol{x}}$, the loss function is often defined to be $L := \frac{1}{2}(\boldsymbol{u}_N - \hat{\boldsymbol{x}})^{\mathrm{T}}(\boldsymbol{u}_N - \hat{\boldsymbol{x}})$; but really any differentiable function of $\boldsymbol{u}_N$ will do.

Derivation. Let each layer in the network perform the same operation, up to some parameters $\mathbf{W}$, $\boldsymbol{b}$, on its inputs: an affine function of those inputs followed by a(n element-wise) nonlinear operation. In symbols, the output of the n^{th} layer, $\boldsymbol{u}_n$, is produced by:

$$\begin{aligned} \boldsymbol{v}_n &= \mathbf{W}_n \boldsymbol{u}_{n-1} + \boldsymbol{b}_n, \\ \boldsymbol{u}_n &= \mathbf{f}_n(\boldsymbol{v}_n), \end{aligned} \tag{5.16}$$

where $\mathbf{f}_n$ is the nonlinearity. We index each "weight matrix" $\mathbf{W}_n$ by the layer it *terminates on.*

We start our derivation, with a little bit of foresight, by defining the backwards recursion:

$$\boldsymbol{g}_N = \mathbf{J}_N^{\mathrm{T}} \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}_N}, \qquad \boldsymbol{g}_{n-1} := \mathbf{J}_{n-1}^{\mathrm{T}} \mathbf{W}_n^{\mathrm{T}} \boldsymbol{g}_n, \text{ for } n = N, N-1, \ldots, 1. \tag{5.17}$$

Here, $\mathbf{J}_n := \mathrm{d}\boldsymbol{u}_n / \mathrm{d}\boldsymbol{v}_n^{\mathrm{T}} = \mathrm{d}\mathbf{f}_n / \mathrm{d}\boldsymbol{v}_n^{\mathrm{T}}$, the Jacobian of the nonlinearity. With the loss function given above for output targets, $\mathrm{d}L / \mathrm{d}\boldsymbol{u}_N = \boldsymbol{u}_N - \hat{\boldsymbol{x}}$. Now we take derivatives, making frequent use of the matrix-calculus results reviewed in Section A.1. Beginning with the output layer:

$$\frac{\mathrm{d}L}{\mathrm{d}\mathbf{W}_N} = \mathbf{J}_N^{\mathrm{T}} (\boldsymbol{u}_N - \hat{\boldsymbol{x}}) \boldsymbol{u}_{N-1}^{\mathrm{T}} = \boldsymbol{g}_N \boldsymbol{u}_{N-1}^{\mathrm{T}}.$$

Proceeding to the second layer, we find:

$$\frac{\mathrm{d}L}{\mathrm{d}\mathbf{W}_{N-1}} = \mathbf{J}_{N-1}^{\mathrm{T}} \mathbf{W}_N^{\mathrm{T}} \boldsymbol{g}_N \boldsymbol{u}_{N-2}^{\mathrm{T}} = \boldsymbol{g}_{N-1} \boldsymbol{u}_{N-2}^{\mathrm{T}}.$$

Proceeding backwards to earlier layers continues the pattern. The derivatives with respect to the biases are a simple variation: substituting $\mathrm{d}\boldsymbol{v}_n / \mathrm{d}\boldsymbol{b}_n = I$ for $\mathrm{d}\boldsymbol{v}_n / \mathrm{d}\mathbf{W}_n = \boldsymbol{u}_n^{\mathrm{T}}$, we see that:

$$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{b}_n} = \boldsymbol{g}_n.$$

Weight changes are made proportional to the *average* error derivatives under the distribution of input/output pairs, $p(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}})$, so the entire procedure can be summarized as:

$$\Delta \mathbf{W}_n \propto \left\langle \boldsymbol{g}_n \boldsymbol{u}_{n-1}^{\mathrm{T}} \right\rangle_p, \qquad \Delta \boldsymbol{b}_n \propto \langle \boldsymbol{g}_n \rangle_p, \tag{5.18}$$

along with Eq. 5.17 and the definition of $\mathbf{J}_n$. Notice, in this connection, that if $\mathbf{f}_n$ is taken (as usual) to operate element-wise, $\mathbf{f}_n(\boldsymbol{v}_n) = [\mathrm{f}_1(v_n^1), \mathrm{f}_2(v_n^2), \ldots, \mathrm{f}_k(v_n^k)]^{\mathrm{T}}$, then $\mathbf{J}_n$ is diagonal, which simplifies the calculation in Eq. 5.17 (it can be performed with a Hadamard product rather than a full matrix multiply).

A statistical view. Although this neural network is entirely deterministic, the learning procedure can be given a statistical interpretation by translating the loss into a probability distribution. A useful (and standard) way to do this is to identify the loss with a(n average) negative log probability; for example, for the usual neural-network loss function, $L = \frac{1}{2}(\boldsymbol{u}_N - \hat{\boldsymbol{x}})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{u}_N - \hat{\boldsymbol{x}})$, we have:

$$\begin{aligned} & q(\hat{\boldsymbol{x}} | \hat{\boldsymbol{y}}; \boldsymbol{\theta}) \propto \exp \left\{ -\frac{1}{2} (\boldsymbol{u}_N(\hat{\boldsymbol{y}}) - \hat{\boldsymbol{x}})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{u}_N(\hat{\boldsymbol{y}}) - \hat{\boldsymbol{x}}) \right\} \\ \implies & q(\hat{\boldsymbol{x}} | \hat{\boldsymbol{y}}; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{u}_N(\hat{\boldsymbol{y}}), \boldsymbol{\Sigma}), \end{aligned}$$

where we have made explicit the dependence of network's output layer on its inputs. The parameters $\boldsymbol{\theta}$ are the weights ($\mathbf{W}_n$) and biases ($\boldsymbol{b}_n$). Although we have introduced a covariance matrix ($\boldsymbol{\Sigma}$) into the equation, it plays no role in the minimization, as can be seen by setting the derivative of the negative log probability to zero. As usual, we exploit the fact that the logarithm is monotonically increasing (its derivative is strictly positive) to see that maximizing $q(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}};\boldsymbol{\theta})$ is equivalent to minimizing the negative log probability, i.e., the loss function. More precisely, we attempt to maximize the *average* of $q(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}};\boldsymbol{\theta})$ (or $\log q(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}};\boldsymbol{\theta})$) under the "data" distribution of inputs and outputs. This maximization can also be re-described as follows:

$$\begin{aligned}
\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\Big\{\langle q(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}};\boldsymbol{\theta})\rangle_{p(\hat{\boldsymbol{y}},\hat{\boldsymbol{x}})}\Big\} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\Big\{\langle -\log q(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}};\boldsymbol{\theta})\rangle_{p(\hat{\boldsymbol{y}},\hat{\boldsymbol{x}})}\Big\} \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\Big\{\langle \log p(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}})\rangle_{p(\hat{\boldsymbol{y}},\hat{\boldsymbol{x}})} - \langle \log q(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}};\boldsymbol{\theta})\rangle_{p(\hat{\boldsymbol{y}},\hat{\boldsymbol{x}})}\Big\} \\
&\approx \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\big\{\mathbb{E}_{p(\hat{\boldsymbol{y}},\hat{\boldsymbol{x}})}[\log p(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}})] - \mathbb{E}_{p(\hat{\boldsymbol{y}},\hat{\boldsymbol{x}})}[\log q(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}};\boldsymbol{\theta})]\big\} \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\{\mathrm{D}_{\mathrm{KL}}\{p(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}})||q(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}};\boldsymbol{\theta})\}\},
\end{aligned}$$

where the quantity being minimized in the final line is the *conditional* KL divergence (or "conditional relative entropy").

So we see that backprop admits of a few, closely related statistical interpretations. It maximizes the average (under the data distribution, $p(\hat{\boldsymbol{y}},\hat{\boldsymbol{x}})$) conditional probability ($q(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}};\boldsymbol{\theta})$) of the network's outputs. This is the same as maximizing the average log probability—or what is usually called the "expected log likelihood," with $q(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}};\boldsymbol{\theta})$ interpreted as a likelihood function for the *parameters*, $\boldsymbol{\theta}$. Or again, it minimizes the (approximate) cross entropy, $\langle -\log q(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}};\boldsymbol{\theta})\rangle_{p(\hat{\boldsymbol{y}},\hat{\boldsymbol{x}})}$. Finally, this is equivalent to implementing a form of conditional density estimation, making the "model conditional density," $q(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}};\boldsymbol{\theta})$, more like the "data conditional density," $p(\hat{\boldsymbol{x}}|\hat{\boldsymbol{y}})$, on average across the distribution $p(\hat{\boldsymbol{y}},\hat{\boldsymbol{x}})$ of inputs and outputs.

Note that the model—the neural network—is only probabilistic at the very end, where data are generated normally about its deterministically computed outputs, $\boldsymbol{u}_N$. Different choices for this output distribution correspond to different loss functions. Alternatively, we can view this noise as allowing for mismatch between the neural network and the true distribution of input/output pairs.

Some examples.

Backpropagation through time (BPTT)

Suppose we have reason to believe that some of the structure of our network is "conserved" or repeated, as in Fig. **??**. For instance, we might imagine segments of the network to correspond to slices of time, in which case it is reasonable to treat every time slice as identical in structure, even though the inputs and outputs change over time. Or we might have reason to believe that certain spatial structures are repeated. In either case, the concrete meaning of the assumption is that the same parameters show up in multiple places in the network. The network is then called "recurrent." The upshot for backprop is that a little care must be taken in distinguishing partial from total derivatives, after which the algorithm goes through as in the previous section.

We consider a generic, deterministic, recurrent neural network (RNN) with inputs $\hat{\boldsymbol{Y}}_1, \ldots, \hat{\boldsymbol{Y}}_T \sim p(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_T)$ and "hidden"-unit activities $\boldsymbol{u}_t$ recursively computed as:

$$\boldsymbol{u}_t = \mathbf{f}(\boldsymbol{u}_{t-1}, \hat{\boldsymbol{y}}_t, \boldsymbol{\theta}). \tag{5.19}$$

For simplicity, we let the recurrent layer also be the output layer—i.e., the layer at which the loss function is evaluated—in which case the most general loss can be expressed as $L(\boldsymbol{u}_0, \ldots, \boldsymbol{u}_T, \boldsymbol{\theta})$. Notice that we let it depend on the hidden units at every time step, as well as directly on the parameters. To compute parameter changes, we take the (total) derivative of the loss function with respect to those parameters:

$$\begin{aligned} \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{\theta}^{\mathrm{T}}} &= \frac{\partial L}{\partial \boldsymbol{\theta}^{\mathrm{T}}} + \sum_{t=0}^{T} \frac{\partial L}{\partial \boldsymbol{u}_t{}^{\mathrm{T}}} \frac{\mathrm{d}\boldsymbol{u}_t}{\mathrm{d}\boldsymbol{\theta}^{\mathrm{T}}} \\ &= \frac{\partial L}{\partial \boldsymbol{\theta}^{\mathrm{T}}} + \sum_{t=0}^{T} \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}_t{}^{\mathrm{T}}} \frac{\partial \boldsymbol{u}_t}{\partial \boldsymbol{\theta}^{\mathrm{T}}}. \end{aligned} \tag{5.20}$$

The first equality is just standard calculus; the second follows because $\boldsymbol{\theta}$ only occurs in the final terms in the chains of derivatives. Thus, one can either compute the total effect of $\boldsymbol{\theta}$ on each $\boldsymbol{u}_t$, and then compute the direct effects of the latter on L; or one can compute the *direct* effect of $\boldsymbol{\theta}$ on each $\boldsymbol{u}_t$, and then compute the *total* effect of the latter on L.

Because of Eq. 5.19, the effect on L of $\boldsymbol{u}_t$, the current hidden state, is its direct effect, plus the effects caused by its influence on *future* hidden states, $\boldsymbol{u}_{t+k}, k > 0$. This can be expressed recursively as:

$$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}_t{}^{\mathrm{T}}} = \frac{\partial L}{\partial \boldsymbol{u}_t{}^{\mathrm{T}}} + \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}_{t+1}{}^{\mathrm{T}}} \frac{\partial \boldsymbol{u}_{t+1}}{\partial \boldsymbol{u}_t{}^{\mathrm{T}}}. \tag{5.21}$$

The recursion is initialized at $\mathrm{d}L/\,\mathrm{d}\boldsymbol{u}_T{}^{\mathrm{T}} = \partial L/\partial \boldsymbol{u}_T{}^{\mathrm{T}}$ and run backwards in time.

The backprop-through time algorithm consists of Eqs. 5.20 and 5.21. Applying it to a particular model requires only working out three partial derivatives, $\partial \boldsymbol{u}_{t+1}/\partial \boldsymbol{u}_t{}^{\mathrm{T}}$, $\partial L/\partial \boldsymbol{u}_t{}^{\mathrm{T}}$, and $\partial L/\partial \boldsymbol{\theta}^{\mathrm{T}}$, for a particular choice of recurrent function and loss function. Using the standard recurrent function for a neural network (cf. Eq. 5.16):

$$\boldsymbol{u}_t = \mathbf{f}(\mathbf{W}_{uu}\boldsymbol{u}_{t-1} + \mathbf{W}_{\hat{y}\hat{x}}\hat{\boldsymbol{y}}_t + \boldsymbol{b}_u),$$

and applying Eqs. 5.20 and 5.21 yields:

$$\begin{aligned} \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}_t{}^{\mathrm{T}}} &= \frac{\partial L}{\partial \boldsymbol{u}_t{}^{\mathrm{T}}} + \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}_{t+1}{}^{\mathrm{T}}} \mathbf{J}_{t+1}\mathbf{W}_{uu}; \\ \frac{\mathrm{d}L}{\mathrm{d}\mathbf{W}_{uu}} &= \frac{\partial L}{\partial \mathbf{W}_{uu}} + \sum_{t=0}^{T} \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}_t{}^{\mathrm{T}}} \frac{\partial \boldsymbol{u}_t}{\partial \mathbf{W}_{uu}} = \frac{\partial L}{\partial \mathbf{W}_{uu}} + \sum_{t=0}^{T} \mathbf{J}_t^{\mathrm{T}} \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}_t} \boldsymbol{u}_{t-1}{}^{\mathrm{T}}, \\ \frac{\mathrm{d}L}{\mathrm{d}\mathbf{W}_{\hat{y}\hat{x}}} &= \frac{\partial L}{\partial \mathbf{W}_{\hat{y}\hat{x}}} + \sum_{t=0}^{T} \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}_t{}^{\mathrm{T}}} \frac{\partial \boldsymbol{u}_t}{\partial \mathbf{W}_{\hat{y}\hat{x}}} = \frac{\partial L}{\partial \mathbf{W}_{\hat{y}\hat{x}}} + \sum_{t=0}^{T} \mathbf{J}_t^{\mathrm{T}} \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}_t} \hat{\boldsymbol{y}}_t{}^{\mathrm{T}}, \\ \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{b}_u{}^{\mathrm{T}}} &= \frac{\partial L}{\partial \boldsymbol{b}_u{}^{\mathrm{T}}} + \sum_{t=0}^{T} \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}_t{}^{\mathrm{T}}} \frac{\partial \boldsymbol{u}_t}{\partial \boldsymbol{b}_u{}^{\mathrm{T}}} = \frac{\partial L}{\partial \boldsymbol{b}_u{}^{\mathrm{T}}} + \sum_{t=0}^{T} \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}_t{}^{\mathrm{T}}} \mathbf{J}_t. \end{aligned} \tag{5.22}$$

Note that the subscript on the Jacobian of the nonlinearity, $\mathbf{J}_t = f'$, indicates the time index of its *output*.

Applying backprop-through-time to the RTRBM. To get a better feel for the algorithm, we consider the recurrent temporal restricted Boltzmann machine (RTRBM), introduced in Section 3.4.1. Only part of the update equations will be derivable with the tools we have seen so far; we shall revisit the remainder once we have seen the methods of unsupervised learning (Sections 6.8.1 and 6.8.2). This complexity is due to the fact that the RTRBM's loss function is not, as we saw in Section 3.4.1, simply a squared-error penalty on outputs—but, as we have lately observed, backprop can be applied (at least in principle) to any differentiable function of the outputs. The RTRBM's loss, in particular, is defined to be:

$$L = \sum_{t=0}^{T} L_t(\boldsymbol{u}_t, \boldsymbol{\theta}), \qquad L_T := -\log q(\hat{\boldsymbol{y}}_0; \boldsymbol{\theta}) \qquad L_t := -\log q(\hat{\boldsymbol{y}}_{t+1} | \boldsymbol{u}_t; \boldsymbol{\theta}) \;\text{ for all other } t,$$

where for notational compactness, we (confusingly) assigned the first term to the *last* loss, since this allows us to write [25]:

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \mathrm{D}_{\mathrm{KL}}\{p(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_T) || q(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_T; \boldsymbol{\theta})\} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \left\langle \sum_{t=0}^{T} L_t(\boldsymbol{u}_t; \boldsymbol{\theta}) \right\rangle_{p(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_T)}.$$

Thus, improving (decreasing) the loss on the right is equivalent to making the RTRBM a good generative model for the data. And the resulting partial derivatives in Eq. 5.22 decouple over time:

$$\frac{\partial L}{\partial \boldsymbol{\theta}^{\mathrm{T}}} = \sum_{t=0}^{T} \frac{\partial L_t}{\partial \boldsymbol{\theta}^{\mathrm{T}}}. \tag{5.23}$$

Now, under this (unusual) loss function, it is not perhaps obvious how to compute the terms $\partial L_t / \partial \mathbf{W}_{uu}$, $\partial L_t / \partial \mathbf{W}_{\hat{y}\hat{x}}$, and $\partial L_t / \partial \boldsymbol{b}_u$ required by Eq. 5.22. In particular, we recall that $q(\hat{\boldsymbol{y}}_{t+1} | \boldsymbol{u}_t; \boldsymbol{\theta})$ is the marginal distribution over the visible units of an RBM (or EFH), and that each RBM in the RTRBM is defined via the conditional distribution $q(\hat{\boldsymbol{y}}_{t+1}, \hat{\boldsymbol{x}}_{t+1} | \boldsymbol{u}_t; \boldsymbol{\theta})$. Thus, computing the average of these loss-function gradients under the data distribution, $p(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_T)$, is equivalent to making each RBM a good model for these data, but—in contrast to the problems solved by backprop—without knowing the values of certain variables, sc., the "latent" variables of the RBMs, $\hat{\boldsymbol{x}}_{t+1}$. Instead, one has only a model of the joint distribution over $\hat{\boldsymbol{y}}_{t+1}, \hat{\boldsymbol{x}}_{t+1}$, in this case the conditional $q(\hat{\boldsymbol{y}}_{t+1}, \hat{\boldsymbol{x}}_{t+1} | \boldsymbol{u}_t; \boldsymbol{\theta})$. We shall consider this type of learning—"latent-variable density estimation"—in the subsequent sections.

In fact, it turns out that the last unspecified derivative, $\partial L_t / \partial {\boldsymbol{u}_t}^{\mathrm{T}}$ (see Eq. 5.21) will also require unsupervised learning, so we defer its calculation until a later time. At present, for economy of expression, we merely define:

$$\begin{aligned}
\boldsymbol{g}_t^{\mathrm{T}} := \frac{\mathrm{d}L}{\mathrm{d}{\boldsymbol{u}_t}^{\mathrm{T}}} \mathbf{J}_t &= \left(\frac{\partial L}{\partial {\boldsymbol{u}_t}^{\mathrm{T}}} + \frac{\mathrm{d}L}{\mathrm{d}{\boldsymbol{u}_{t+1}}^{\mathrm{T}}} \frac{\partial \boldsymbol{u}_{t+1}}{\partial {\boldsymbol{u}_t}^{\mathrm{T}}} \right) \mathbf{J}_t \\
&= \left(\frac{\partial L}{\partial {\boldsymbol{u}_t}^{\mathrm{T}}} + \frac{\mathrm{d}L}{\mathrm{d}{\boldsymbol{u}_{t+1}}^{\mathrm{T}}} \mathbf{J}_{t+1} \mathbf{W}_{uu} \right) \mathbf{J}_t \\
&= \left(\frac{\partial L}{\partial {\boldsymbol{u}_t}^{\mathrm{T}}} + \boldsymbol{g}_{t+1}^{\mathrm{T}} \mathbf{W}_{uu} \right) \mathbf{J}_t
\end{aligned}$$

where the second equality follows from Eq. 5.21. This constitutes the backward recursion. The gradients, according to which weight changes will be made, can then be written:

$$\begin{aligned}
\frac{\mathrm{d}L}{\mathrm{d}\mathbf{W}_{uu}} &= \sum_{t=0}^{T}\Big(-\frac{\partial \log q(\hat{\boldsymbol{y}}_{t+1}|\boldsymbol{u}_t;\boldsymbol{\theta})}{\partial \mathbf{W}_{uu}} + \boldsymbol{g}_t \boldsymbol{u}_{t-1}{}^{\mathrm{T}}\Big),\\
\frac{\mathrm{d}L}{\mathrm{d}\mathbf{W}_{\hat{y}\hat{x}}} &= \sum_{t=0}^{T}\Big(-\frac{\partial \log q(\hat{\boldsymbol{y}}_{t+1}|\boldsymbol{u}_t;\boldsymbol{\theta})}{\partial \mathbf{W}_{\hat{y}\hat{x}}} + \boldsymbol{g}_t \hat{\boldsymbol{y}}_t{}^{\mathrm{T}}\Big),\\
\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{b}_u{}^{\mathrm{T}}} &= \sum_{t=0}^{T}\Big(-\frac{\partial \log q(\hat{\boldsymbol{y}}_{t+1}|\boldsymbol{u}_t;\boldsymbol{\theta})}{\partial \boldsymbol{b}_u} + \boldsymbol{g}_t^{\mathrm{T}}\Big),\\
\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{b}_{\hat{y}}{}^{\mathrm{T}}} &= \sum_{t=0}^{T} -\frac{\partial \log q(\hat{\boldsymbol{y}}_{t+1}|\boldsymbol{u}_t;\boldsymbol{\theta})}{\partial \boldsymbol{b}_{\hat{y}}{}^{\mathrm{T}}}.
\end{aligned} \tag{5.24}$$

We shall return to these equations in Section 6.8.2.

5.2 Unsupervised learning

So far we have considered discriminative models in which the "outputs," $\hat{\boldsymbol{X}}$, have been observed. Indeed, it might seem rather quixotic to try to learn a discriminative model purely from its inputs, $\hat{\boldsymbol{Y}}$. Nevertheless, let us consider the intuitive objective of maximizing information transmission [1], and see where it leads us.

5.2.1 "InfoMax" in deterministic, invertible models

To motivate the objective, consider the deterministic input-output relationship

$$\hat{\boldsymbol{Z}} = \boldsymbol{f}(\boldsymbol{Y}, \boldsymbol{\theta}). \tag{5.25}$$

(Notice that we have introduced a new symbol for the outputs; this will be justified shortly.) For example, when there are fewer outputs than inputs, $\left|\hat{\boldsymbol{Z}}\right| < |\boldsymbol{Y}|$, changing $\boldsymbol{\theta}$ so as to maximize mutual information between inputs and outputs will yield a function $\boldsymbol{f}(\cdot, \boldsymbol{\theta})$ that recodes its inputs more efficiently. However, we begin with a seemingly more näive, but mathematically tractable, alternative: Let the number of outputs equal the number inputs, $\left|\hat{\boldsymbol{Z}}\right| = |\boldsymbol{Y}|$, and furthermore let $\boldsymbol{f}$ be invertible (in its first argument), so that $\boldsymbol{Y} = \boldsymbol{f}^{-1}(\hat{\boldsymbol{Z}}, \boldsymbol{\theta})$. This may seem somewhat perverse, since invertible transformations are automatically information preserving. However, recall that mutual information nevertheless depends on $\boldsymbol{\theta}$:

$$\mathcal{I}\Big(\boldsymbol{Y}; \hat{\boldsymbol{Z}}\Big) = \mathrm{H}\Big[\hat{\boldsymbol{Z}}; \boldsymbol{\theta}\Big] - \mathrm{H}\Big[\hat{\boldsymbol{Z}} | \boldsymbol{Y}; \boldsymbol{\theta}\Big] = \mathrm{H}\Big[\hat{\boldsymbol{Z}}; \boldsymbol{\theta}\Big]. \tag{5.26}$$

That is, although the conditional entropy is zero for all values of $\boldsymbol{\theta}$ (under the assumption that $\boldsymbol{f}(\cdot, \boldsymbol{\theta})$ doesn't lose its invertibility), the output entropy still depends on $\boldsymbol{\theta}$. In fine, for invertible $\boldsymbol{f}(\cdot, \boldsymbol{\theta})$, maximizing mutual information amounts to maximizing output entropy.[4] Since a vector random variable $\hat{\boldsymbol{Z}}$ is maximally entropic only if its components are statistically independent, this suggests that the "InfoMax" criterion can underwrite a form of *independent-components analysis*. That is, maximizing input-output mutual information

independent-components analysis

[4] If there *were* noise in the map from $\boldsymbol{Y}$ to $\hat{\boldsymbol{Z}}$, this would add an extra term to the gradient, and the goal of maximizing output entropy would have to be balanced against noise-proofing the transmission.

or output entropy will "unmix" the inputs $\hat{\boldsymbol{Y}}$ into their independent components.

Without yet specifying the form of $\boldsymbol{f}$, let us try to re-express this objective, Eq. 5.26. More precisely, we consider its negation, $-\mathrm{H}\Big[\hat{\boldsymbol{Z}};\boldsymbol{\theta}\Big]$, for consistency with the standard objectives of this book, which are losses. To re-express this quantity we note that, although we have not specified a distribution for $\hat{\boldsymbol{Z}}$, $\hat{p}_{\hat{\boldsymbol{Z}}}\Big(\hat{\boldsymbol{Z}}\Big)$ is inherited directly from the data distribution $p_{\boldsymbol{Y}}(\boldsymbol{Y})$ via the deterministic relationship in Eq. 5.25. In particular, since the relationship is (by assumption) invertible, the two distributions are related by the standard change-of-variables formula, which we apply on the third line:

$$\begin{aligned}
-\mathcal{I}\Big(\boldsymbol{Y};\hat{\boldsymbol{Z}}\Big) = -\mathrm{H}\Big[\hat{\boldsymbol{Z}}\Big] &= \mathbb{E}_{\hat{\boldsymbol{Z}}}\Big[\log \hat{p}_{\hat{\boldsymbol{Z}}}\Big(\hat{\boldsymbol{Z}}\Big)\Big] \\
&= \mathbb{E}_{\boldsymbol{Y}}\big[\log \hat{p}_{\hat{\boldsymbol{Z}}}(\boldsymbol{f}(\boldsymbol{Y},\boldsymbol{\theta}))\big] \\
&= \mathbb{E}_{\boldsymbol{Y}}\left[\log \frac{p_{\boldsymbol{Y}}(\boldsymbol{Y})}{\left|\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{y}}(\boldsymbol{Y},\boldsymbol{\theta})\right|}\right] \\
&= \mathrm{D}_{\mathrm{KL}}\left\{p_{\boldsymbol{Y}}(\boldsymbol{Y})\middle|\middle|\left|\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{y}}(\boldsymbol{Y},\boldsymbol{\theta})\right|\right\}.
\end{aligned} \tag{5.27}$$

Evidently, maximizing mutual information through an invertible transformation is identical to minimizing our standard loss, the relative entropy between data and model distributions, with the latter equal to the Jacobian determinant of that transformation:

$$\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) := \left|\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{y}}(\hat{\boldsymbol{Y}},\boldsymbol{\theta})\right|. \tag{5.28}$$

The assimilation to density estimation can be completed by reinterpreting Eq. 5.25 as defining (via its inverse) the emission density of a *generative* model. We defer this reinterpretion until Chapter **??**, when we take up learning in generative models in earnest.

InfoMax independent-components analysis. One interesting special case of the unsupervised, discriminative learning problem just described is so-called InfoMax ICA [1]. Here, the transformation of Eq. 5.25 is conceived as a neural network (see Fig. 5.1), and therefore consisting of two steps: linear transformation followed by elementwise nonlinearity:

$$\hat{\boldsymbol{X}} = \mathbf{G}\,\hat{\boldsymbol{Y}}, \qquad\qquad \hat{Z}_k = \psi_k(\hat{X}_k), \text{ for all } k. \tag{5.29}$$

We leave the output nonlinearity undefined for now, except to insist that it be invertible. It could, for example, be the classic logistic (sigmoid) function. For convenience we also assign the symbol $\boldsymbol{\psi}$ to the vector-valued "stack" of $\psi_k(\hat{X}_k)$ for all k:

$$\{\boldsymbol{\psi}(\hat{\boldsymbol{X}})\}_k := \psi_k(\hat{X}_k). \tag{5.30}$$

The gradient, whether of the mutual information, output entropy, relative input entropy, or input cross entropy, is then

$$\begin{aligned}
-\frac{\mathrm{d}}{\mathrm{d}\mathbf{G}}\mathcal{I}\Big(\boldsymbol{Y};\hat{\boldsymbol{Z}}\Big) = -\frac{\mathrm{d}}{\mathrm{d}\mathbf{G}}\mathrm{H}\Big[\hat{\boldsymbol{Z}}\Big] = \frac{\mathrm{d}}{\mathrm{d}\mathbf{G}}\,\mathrm{D}_{\mathrm{KL}}\left\{p_{\boldsymbol{Y}}(\boldsymbol{Y})\middle|\middle|\left|\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{y}}(\boldsymbol{Y},\mathbf{G})\right|\right\} &= \frac{\mathrm{d}}{\mathrm{d}\mathbf{G}}\mathbb{E}_{\boldsymbol{Y}}\left[-\log\left|\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{y}}(\boldsymbol{Y},\mathbf{G})\right|\right] \\
&= \frac{\mathrm{d}}{\mathrm{d}\mathbf{G}}\mathbb{E}_{\boldsymbol{Y}}\left[-\log\left|\frac{\mathrm{d}\boldsymbol{\psi}}{\mathrm{d}\hat{\boldsymbol{x}}}(\mathbf{G}\,\boldsymbol{Y})\mathbf{G}\right|\right] \\
&= \frac{\mathrm{d}}{\mathrm{d}\mathbf{G}}\mathbb{E}_{\boldsymbol{Y}}\left[-\log\left|\frac{\mathrm{d}\boldsymbol{\psi}}{\mathrm{d}\hat{\boldsymbol{x}}}(\mathbf{G}\,\boldsymbol{Y})\right|\right] - \mathbf{G}^{-\mathrm{T}}.
\end{aligned} \tag{5.31}$$

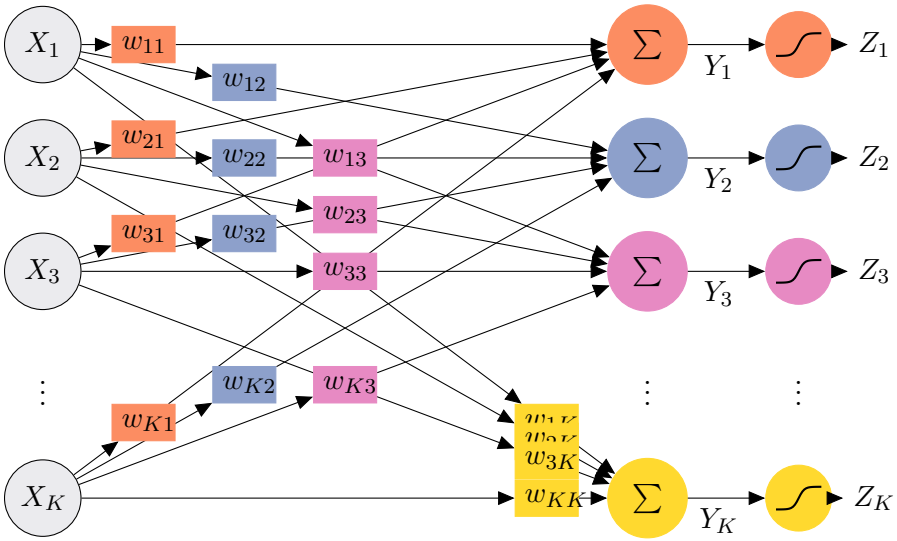


Figure 5.1: **A square neural network.**

[[Why the components are independent; limitations based on linear model; etc.]]

"Semi-supervised" clustering. [[XXX]]

Chapter 6

Learning in Generative Models

6.1 Introduction

[[We are concerned with learning in generative models under two circumstances, "supervised" and "unsupervised." The paradigm case of each is the illustrated in Fig. **??**. In Fig. **??**, each datum ($\boldsymbol{y}$) comes with a class label—but despite the "missing" labels in Fig. **??**, class structure is still perspicuous. Both sets of data can be fit with a Gaussian mixture model (Chapter 2), but in the case of Fig. **??**, the "source" variables $\hat{\boldsymbol{X}}$ are "latent" or unobserved. The learning algorithms for the two cases are consequently different. Nevertheless, for the GMM, the supervised-learning algorithm can be written as a special case of the unsupervised-learning algorithm, and indeed this is often the case. So we shall concentrate on the unsupervised problems, deriving the supervised solutions along the way.]]

Density estimation. Let us begin with an even simpler data set to model, Fig. **??**. The data look to be distributed normally, so it would be sensible simply to let $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$ be $\mathcal{N}(\boldsymbol{m},\mathbf{S})$, with $\boldsymbol{m}$ and $\mathbf{S}$ the sample mean and sample covariance. But let us proceed somewhat naïvely according to the generic procedure introduced in Section 4.2. The procedure enjoins us to minimize the relative entropy; or, equivalently, since the entropy doesn't depend on the parameters, the cross entropy:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{\boldsymbol{Y}}[-\log\hat{p}(\boldsymbol{Y};\boldsymbol{\theta})] \\ &= \mathbb{E}_{\boldsymbol{Y}}\left[-\log\left[\tau^{-M/2}|\boldsymbol{\Sigma}|^{-1/2}\exp\left\{-\frac{1}{2}(\boldsymbol{Y}-\boldsymbol{\mu}))^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y}-\boldsymbol{\mu}))\right\}\right]\right] \\ &= \mathbb{E}_{\boldsymbol{Y}}\left[-\frac{1}{2}\Big[M\log\tau+\log|\boldsymbol{\Sigma}|+(\boldsymbol{Y}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y}-\boldsymbol{\mu})\Big]\right].\end{aligned}$$

Differentiating with respect to $\boldsymbol{\mu}$ indeed indicates that $\boldsymbol{\mu}$ should be set equal to the sample average:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\mu}} &= \mathbb{E}_{\boldsymbol{Y}}\big[\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y}-\boldsymbol{\mu})\big] \stackrel{\text{set}}{=} 0 \\ \implies \boldsymbol{\mu} &= \mathbb{E}_{\boldsymbol{Y}}[\boldsymbol{Y}] \approx \langle\,\boldsymbol{Y}\,\rangle_{\boldsymbol{Y}},\end{aligned}$$

where in the final line we approximate the expectation under the (unavailable) data distribution with an average under (available) samples from it. Likewise, differentiating with

respect to $\boldsymbol{\Sigma}^{-1}$, we find (after consulting Section A.1)

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\Sigma}^{-1}} = \mathbb{E}_{\boldsymbol{Y}}\left[\frac{1}{2}\Big[\boldsymbol{\Sigma} - (\boldsymbol{Y}-\boldsymbol{\mu})(\boldsymbol{Y}-\boldsymbol{\mu})^{\mathrm{T}}\Big]\right] \stackrel{\text{set}}{=} 0$$
$$\implies \boldsymbol{\Sigma} = \mathbb{E}_{\boldsymbol{Y}}\Big[(\boldsymbol{Y}-\boldsymbol{\mu})(\boldsymbol{Y}-\boldsymbol{\mu})^{\mathrm{T}}\Big] \approx \Big\langle (\boldsymbol{Y}-\boldsymbol{\mu})(\boldsymbol{Y}-\boldsymbol{\mu})^{\mathrm{T}}\Big\rangle_{\boldsymbol{Y}}.$$

So far, so good. We now proceed to the dataset shown in Fig. **??**. Here by all appearances is a mixture of Gaussians. In Section 2.1.1 we derived the marginal distribution for the GMM, Eq. 2.10, so it seems that perhaps we can use the same procedure as for the single Gaussian. The cross-entropy loss is

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{\boldsymbol{Y}}[-\log \hat{p}(\boldsymbol{Y};\boldsymbol{\theta})] \\ &= \mathbb{E}_{\boldsymbol{Y}}\left[-\log\left(\sum_{k=1}^{K}\hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{X}_k=1;\boldsymbol{\theta}\Big)\pi_k\right)\right] \\ &= \mathbb{E}_{\boldsymbol{Y}}\left[-\log\left(\sum_{k=1}^{K}\tau^{-M/2}\Big|\boldsymbol{\Sigma}_k\Big|^{-1/2}\exp\Big\{-\frac{1}{2}\Big(\boldsymbol{Y}-\boldsymbol{\mu}_k)\Big)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}\Big(\boldsymbol{Y}-\boldsymbol{\mu}_k)\Big)\Big\}\pi_k\right)\right].\end{aligned} \tag{6.1}$$

We have encountered a problem. The summation (across classes) inside the logarithm couples the parameters of the K Gaussians together. In particular, differentiating with respect to $\boldsymbol{\mu}_k$ or $\boldsymbol{\Sigma}_k$ will yield an expression involving *all* the means and covariances. Solving these coupled equations (after setting the gradients to zero) is not straightforward.

6.2 Latent-variable density estimation

Now notice that the root of the problem, the summation sign in Eq. 6.1, was introduced by the marginalization. Indeed, the joint distribution, the product of Eqs. 2.3 and 2.4, is

$$\hat{p}\Big(\hat{\boldsymbol{X}},\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) = \prod_{k}^{K}\Big[\mathcal{N}\Big(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\Big)\pi_k\Big]^{\hat{X}_k}. \tag{6.2}$$

The log of this distribution evidently does decouple into a sum of K terms, each involving only $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, and π_k So perhaps we should try to re-express the marginal cross entropy, or anyway its gradient, in terms of the joint cross entropy.

Introducing the joint distribution. Starting with the gradient of a generic marginal cross entropy, we move the derivative into the expection, "anti-marginalize" to restore the

latent variables, and then rearrange terms:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{Y}}[-\log\hat{p}(\,\boldsymbol{Y};\boldsymbol{\theta})] &= \mathbb{E}_{\boldsymbol{Y}}\left[-\frac{1}{\hat{p}(\,\boldsymbol{Y};\boldsymbol{\theta})}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\hat{p}(\,\boldsymbol{Y};\boldsymbol{\theta})\right] \\
&= \mathbb{E}_{\boldsymbol{Y}}\left[-\frac{1}{\hat{p}(\,\boldsymbol{Y};\boldsymbol{\theta})}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\sum_{\hat{\boldsymbol{x}}}\hat{p}(\hat{\boldsymbol{x}},\,\boldsymbol{Y};\boldsymbol{\theta})\right] \\
&= \mathbb{E}_{\boldsymbol{Y}}\left[-\frac{1}{\hat{p}(\,\boldsymbol{Y};\boldsymbol{\theta})}\sum_{\hat{\boldsymbol{x}}}\hat{p}(\hat{\boldsymbol{x}},\,\boldsymbol{Y};\boldsymbol{\theta})\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\log\hat{p}(\hat{\boldsymbol{x}},\,\boldsymbol{Y};\boldsymbol{\theta})\right] \\
&= \mathbb{E}_{\boldsymbol{Y}}\left[-\sum_{\hat{\boldsymbol{x}}}\hat{p}(\hat{\boldsymbol{x}}\,|\,\boldsymbol{Y};\boldsymbol{\theta})\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\log\hat{p}(\hat{\boldsymbol{x}},\,\boldsymbol{Y};\boldsymbol{\theta})\right] \\
&= \mathbb{E}_{\boldsymbol{Y}}\left[\mathbb{E}_{\hat{\boldsymbol{X}}\,|\,\boldsymbol{Y}}\left[-\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\log\hat{p}\Big(\hat{\boldsymbol{X}},\,\boldsymbol{Y};\boldsymbol{\theta}\Big)\,\Big|\,\boldsymbol{Y}\right]\right] \\
&= \mathbb{E}_{\hat{\boldsymbol{X}},\boldsymbol{Y}}\left[-\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\log\hat{p}\Big(\hat{\boldsymbol{X}},\,\boldsymbol{Y};\boldsymbol{\theta}\Big)\right],
\end{aligned} \tag{6.3}$$

where in the final line we have combined the data distribution and the model recognition distribution into a single *hybrid joint distribution*, $\hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\boldsymbol{Y};\boldsymbol{\theta}\Big)p(\,\boldsymbol{Y})$. This looks promising. It *almost* says that the gradient of the marginal (or "incomplete") cross entropy is the same as the gradient of a joint (or "complete") cross entropy. But the derivative cannot pass outside the expectation, because the hybrid joint distribution depends, like the model distribution, on the parameters $\boldsymbol{\theta}$.

hybrid joint distribution

To see the implications of this dependence on the parameters, we return to our workhorse example, the Gaussian mixture model. Inserting Eq. 6.2 into the final line of Eq. 6.3 shows that

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{Y}}[-\log\hat{p}(\,\boldsymbol{Y};\boldsymbol{\theta})] &= \mathbb{E}_{\hat{\boldsymbol{X}},\boldsymbol{Y}}\left[-\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\log\hat{p}\Big(\hat{\boldsymbol{X}},\,\boldsymbol{Y};\boldsymbol{\theta}\Big)\right] \\
&= \mathbb{E}_{\hat{\boldsymbol{X}},\boldsymbol{Y}}\left[-\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\log\prod_{k}^{K}\Big[\mathcal{N}\Big(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\Big)\pi_k\Big]^{\hat{X}_k}\right] \\
&= \mathbb{E}_{\hat{\boldsymbol{X}},\boldsymbol{Y}}\left[\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\sum_{k}^{K}\hat{X}_k\left(M\log\tau+\log\Big|\boldsymbol{\Sigma}_k\Big|+\Big(\,\boldsymbol{Y}-\boldsymbol{\mu}_k\Big)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}\Big(\,\boldsymbol{Y}-\boldsymbol{\mu}_k\Big)+\log\pi_k\right)\right].
\end{aligned}$$

The gradient with respect to (e.g.) $\boldsymbol{\mu}_k$ is therefore

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\mu}_k}\mathbb{E}_{\boldsymbol{Y}}[-\log\hat{p}(\,\boldsymbol{Y};\boldsymbol{\theta})] &= \mathbb{E}_{\hat{\boldsymbol{X}},\boldsymbol{Y}}\left[\frac{1}{2}\hat{X}_k\boldsymbol{\Sigma}^{-1}\Big(\,\boldsymbol{Y}-\boldsymbol{\mu}_k\Big)\right]\overset{\text{set}}{=}0 \\
&\implies \boldsymbol{\mu}_k = \frac{\mathbb{E}_{\hat{\boldsymbol{X}},\boldsymbol{Y}}\Big[\hat{X}_k\,\boldsymbol{Y}\Big]}{\mathbb{E}_{\hat{\boldsymbol{X}},\boldsymbol{Y}}\Big[\hat{X}_k\Big]}.
\end{aligned} \tag{6.4}$$

This formula looks elegant only if we forget that $\boldsymbol{\mu}_k$ is on both sides. The expectations under the model recognition distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\boldsymbol{Y};\boldsymbol{\theta}\Big)$, involve $\boldsymbol{\mu}_k$—indeed, they involve

all of the parameters (recall Eq. 2.6)! This is reminiscent of the problem with the direct optimization of the marginal cross entropy, Eq. 6.1, so it seems perhaps we have made no progress.

6.3 Expectation-Maximization

Approximating the recognition distribution. This isn't quite true. Working with the joint did allow the parameters *inside* the expectation to decouple. The present problem is a consequence of the parameters in the expectation itself, in particular the model recognition distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\boldsymbol{Y};\boldsymbol{\theta}\Big)$. This does not preclude gradient descent, and we shall return to it and Eq. 6.3 in Section 6.8.1.

But here we will not yet give up hope of a cleaner solution. Consider this seemingly artless proposal: simply take the expectation under a different distribution, $\check{p}\big(\check{\boldsymbol{X}}\big|\boldsymbol{Y}\big)$, one that doesn't depend on these parameters.[1] Doing so will certainly ruin the equality in Eq. 6.3; that is,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{Y}}[-\log\hat{p}(\boldsymbol{Y};\boldsymbol{\theta})] &= \mathbb{E}_{\hat{\boldsymbol{X}},\boldsymbol{Y}}\left[-\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\log\hat{p}\Big(\hat{\boldsymbol{X}},\boldsymbol{Y};\boldsymbol{\theta}\Big)\right]\\ &\neq \mathbb{E}_{\check{\boldsymbol{X}},\boldsymbol{Y}}\left[-\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\log\hat{p}\big(\check{\boldsymbol{X}},\boldsymbol{Y};\boldsymbol{\theta}\big)\right] = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\mathbb{E}_{\check{\boldsymbol{X}},\boldsymbol{Y}}\big[-\log\hat{p}\big(\check{\boldsymbol{X}},\boldsymbol{Y};\boldsymbol{\theta}\big)\big],\end{aligned}$$

where in the bottom lines the expectation is under this *proxy recognition distribution*. But, as indicated by the final equality, and in contrast to the correct gradient, this incorrect gradient is indeed the gradient of a joint ("complete") cross entropy: the derivative can pass outside the expectation that no longer depends on $\boldsymbol{\theta}$.

proxy recognition distribution

The central intuition behind the algorithms that follow is to fit marginal densities by minimizing the joint or complete cross entropy, $\mathbb{E}_{\check{\boldsymbol{X}},\boldsymbol{Y}}\big[-\log\hat{p}\big(\check{\boldsymbol{X}},\boldsymbol{Y};\boldsymbol{\theta}\big)\big]$ in lieu of the marginal or incomplete cross entropy, $\mathbb{E}_{\boldsymbol{Y}}[-\log\hat{p}(\boldsymbol{Y};\boldsymbol{\theta})]$. The discrepancy between these cross entropies can be finessed (we claim) by simultaneously minimizing the discrepancy between our proxy recognition distribution and the true (model) recognition distribution.

Relative entropy, cross entropy, and free energy. Let us prove this last claim. We begin by noting the equivalence of the following optimizations:

$$\begin{aligned}\operatorname*{argmin}_{\boldsymbol{\theta}}\Big\{\mathrm{D}_{\mathrm{KL}}\big\{\check{p}\big(\check{\boldsymbol{X}}\big|\boldsymbol{Y}\big)p(\boldsymbol{Y})\big\|\hat{p}\big(\check{\boldsymbol{X}},\boldsymbol{Y};\boldsymbol{\theta}\big)\big\}\Big\} &= \operatorname*{argmin}_{\boldsymbol{\theta}}\Big\{\mathbb{E}_{\check{\boldsymbol{X}},\boldsymbol{Y}}\big[\log\big(\check{p}\big(\check{\boldsymbol{X}}\big|\boldsymbol{Y}\big)p(\boldsymbol{Y})\big) - \log\hat{p}\big(\check{\boldsymbol{X}},\boldsymbol{Y};\boldsymbol{\theta}\big)\big]\Big\}\\ &= \operatorname*{argmin}_{\boldsymbol{\theta}}\Big\{\mathbb{E}_{\check{\boldsymbol{X}},\boldsymbol{Y}}\big[\log\check{p}\big(\check{\boldsymbol{X}}\big|\boldsymbol{Y}\big) - \log\hat{p}\big(\check{\boldsymbol{X}},\boldsymbol{Y};\boldsymbol{\theta}\big)\big]\Big\}\\ &= \operatorname*{argmin}_{\boldsymbol{\theta}}\Big\{\mathbb{E}_{\check{\boldsymbol{X}},\boldsymbol{Y}}\big[-\log\hat{p}\big(\check{\boldsymbol{X}},\boldsymbol{Y};\boldsymbol{\theta}\big)\big]\Big\}.\end{aligned} \tag{6.5}$$

Although minimized by the same settings of the parameters, the three optimized quantities are all different:

[1] Note the new latent variables $\check{\boldsymbol{X}}$, as opposed to $\hat{\boldsymbol{X}}$, that have necessarily been introduced with this proxy recognition distribution. This distinction is not always noted in the literature.

- The first optimized quantity is a relative entropy, in this case between a "hybrid" joint distribution—the product of the proxy recognition distribution and the data distribution—and the model joint. Thus our optimization can be expressed in terms of the fundamental loss function proposed in Section 4.2.

- The second optimized quantity we call the *free energy*, $\mathcal{F}$, after its counterpart in statistical physics. It differs from the relative entropy only by the entropy of the data. The proof is most elegantly expressed in terms of the free energy and accordingly will be presented that way. *free energy*

- The quantity in the third line is the joint cross entropy that we lately proposed to optimize in lieu of the marginal cross entropy. We will construct our optimization algorithms in terms of the joint cross entropy. Notice that the argmin of the cross entropy would not be equal to the other two if the expectation were taken under the true recognition distribution, because the entropy of the recognition distribution would then depend on the parameters $\boldsymbol{\theta}$.

We start our proof by rearranging the free energy:

$$\begin{aligned}\mathcal{F}(\boldsymbol{\theta}, \check{p}) &:= \mathbb{E}_{\check{\boldsymbol{X}}, \boldsymbol{Y}}\big[\log \check{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}\big) - \log \hat{p}\big(\check{\boldsymbol{X}}, \boldsymbol{Y}; \boldsymbol{\theta}\big)\big] \\ &= \mathbb{E}_{\boldsymbol{Y}}[-\log \hat{p}(\boldsymbol{Y}; \boldsymbol{\theta})] + \mathbb{E}_{\check{\boldsymbol{X}}, \boldsymbol{Y}}\big[\log \check{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}\big) - \log \hat{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}; \boldsymbol{\theta}\big)\big] \\ &= \mathrm{H}_{p\hat{p}}[\boldsymbol{Y}; \boldsymbol{\theta}] + \mathrm{D}_{\mathrm{KL}}\big\{\check{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}\big) \big|\big| \hat{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}; \boldsymbol{\theta}\big)\big\}.\end{aligned} \tag{6.6}$$

The first term is the marginal cross entropy—the quantity that we actually want to minimize. The second term is non-negative, because it is a relative entropy (see Section 4.2). So the free energy is an upper bound on the marginal cross entropy. Given the equivalence of the optimizations in Eq. 6.5, this shows that minimizing the joint cross entropy (via $\boldsymbol{\theta}$) lowers an upper bound on the marginal cross entropy. And the bound can be tightened by optimizing the free energy with respect to the recognition distribution, i.e., decreasing the relative entropy term.

Let us make the procedure explicit. We minimize the free energy with respect to its two arguments:

EM Algorithm

- **(E) Discriminative optimization**: $\check{p}_{t+1}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}\big) \overset{\text{set}}{=} \operatorname{argmin}_{\check{p}}\{\mathcal{F}(\boldsymbol{\theta}_t, \check{p})\}$
- **(M) Generative optimization**: $\boldsymbol{\theta}_{t+1} \overset{\text{set}}{=} \operatorname{argmin}_{\boldsymbol{\theta}}\{\mathcal{F}(\boldsymbol{\theta}, \check{p}_{t+1})\}$.

The parameters can be initialized randomly, $\boldsymbol{\theta}_0$, although there are better alternatives (see below). The two optimizations may be executed alternately or simultaneously—we shall see examples of both below—until convergence, at which point $\boldsymbol{\theta}_{\text{final}}$ is our solution.

What remains to be specified now is a form for the proxy recognition distribution and (in turn) a method for optimizing it. These choices lead to different algorithms. In this text, we refer to all such algorithms as *expectation-maximization* (EM). In fact, only one special case (discussed next) corresponds to the original EM algorithm of Dempster and Laird [**?**], but it was later generalized to the procedure just described under the same name, by Neal and Hinton [17]. *expectation-maximization*

What's in a name? The E and M in the "EM algorithm" originally stood for the two optimization steps that we have called "discriminative" and "generative." Historically, the discriminative optimization was known as the "E step" because it referred to *taking expectations under*, rather than finding via optimization, the proxy posterior distribution. This is the expectation in the final line of Eq. 6.5 (ignoring the additional average under the data distribution). Indeed, as we shall see shortly, the process of computing the expectations can be rather involved, since it requires running an inference algorithm (recall, for example, the smoothers of Section 2.2). Nevertheless, the "E step" has come to refer to the optimization, as this plays a larger role in modern variants of EM—and this allows us to see EM as a form of coordinate descent in the free energy.

Since EM was originally written in terms of expected log-likelihood rather than cross entropy, the "M step" originally referred to a maximization, but it will equally well refer to a minimization as in our setup.

6.4 Learning with exact inference

We saw in Chapter 2 that computing the recognition distribution is frequently impossible, because computing the normalizer $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$ requires either an intractable integral or a sum over an exponential number of terms (i.e., exponential in the number of configurations of the variables to be marginalized out, $\hat{\boldsymbol{X}}$). Nevertheless, we begin with models for which exact inference is tractable, where the algorithm is particularly elegant. Such models are sometimes described as *invertible*.

invertible generative models

In this classical version of EM, the two free-energy optimizations are carried out in consecutive steps. At discriminative step t, the optimization is trivial:

$$\begin{aligned}\check{p}_{t+1}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y}\big) &= \operatorname*{argmin}_{\check{p}}\big\{\mathrm{H}_{p\hat{p}}[\,\boldsymbol{Y};\boldsymbol{\theta}_t] + \mathrm{D}_{\mathrm{KL}}\big\{\check{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y}\big)\big\|\hat{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y};\boldsymbol{\theta}_t\big)\big\}\big\}\\ &= \hat{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y};\boldsymbol{\theta}_t\big).\end{aligned}$$

The conclusion follows because the first term (the marginal cross entropy) doesn't depend on the proxy recognition distribution; because the second term, the relative entropy, is minimal (0) when its arguments are equal; and because we have assumed the model recognition distribution is computable, and therefore an available choice for the proxy recognition distribution. Thus the algorithm becomes:

EM Algorithm under exact inference

- **E step**: $\quad \check{p}_{t+1}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y}\big) \overset{\text{set}}{=} \hat{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y};\boldsymbol{\theta}_t\big)$
- **M step**: $\quad \boldsymbol{\theta}_{t+1} \overset{\text{set}}{=} \operatorname{argmin}_{\boldsymbol{\theta}}\{\mathcal{F}(\boldsymbol{\theta},\check{p}_{t+1})\}.$

That is, we will carry out the optimization in Eq. 6.5 with an iterative process, at each iteration of which we take expectations under the model recognition distribution from the previous iteration. For example, to update our estimate of the mean of the GMM, we will indeed use Eq. 6.4, except that the recognition distribution under which the expectations are taken will be evaluated at the parameters from the previous iteration. This eliminates the dependence on $\boldsymbol{\mu}_k$ from the right-hand side of the equation.

As we saw in the informal proof in the previous section, this procedure is guaranteed to decrease (or, more precisely, not to increase) an upper bound on the loss we actually care about, the marginal cross entropy. In this version of the algorithm, we can say more. At each E step, the relative-entropy term in Eq. 6.6 is not merely reduced but eliminated. So at the start of every M step, the bound is tight (Fig. **??**). That means that any decrease in free energy at this point entails a decrease in marginal cross entropy—a better model for the data. Typically this decrease in free energy will also be accompanied by an increase in the relative-entropy term. But so far from being a bad thing, this corresponds to an even larger decrease in marginal cross entropy than the decrease in free energy (see again Fig. **??**). And at the next step E step, the relative entropy is again eliminated—the bound is again made tight.

In the examples we consider next, the M step is carried out in closed form, so every parameter update either decreases the marginal cross entropy or does nothing at all. In contrast, for models in which the M step is carried out by gradient descent, a decrease in free energy need not correspond to a decrease in marginal cross entropy: As the free energy decreases across the course of the M step, the relative entropy can open back up—the bound can loosen—and therefore further decreases can correspond to the bound tightening again, rather than the marginal cross entropy decreasing. Indeed, the marginal cross entropy can *increase* during this period of bound tightening. But it can never increase above its value at at the beginning of the M step, when the bound was tight.

Applying EM. The EM "algorithm" is a in fact a kind of meta-algorithm or recipe for estimating densities with latent variables. To derive actual algorithms we need to apply EM to specific graphical models. In the following sections we apply EM to some of the most classic latent-variable models.

As usual, we attempt to minimize cross entropies. However, to keep the derivation general, we will avoid specifying the averaging distribution until as late as possible, simply calling it "$\check{p}$." Thus, *the following derivations apply equally well to fully observed models*, in which case

$$\check{p}\big(\check{\boldsymbol{X}},\, \boldsymbol{Y}\big) = p(\boldsymbol{X},\, \boldsymbol{Y}),$$

as they do to a single step in EM, in which case

$$\check{p}\big(\check{\boldsymbol{X}},\, \boldsymbol{Y}\big) = \hat{p}\Big(\hat{\boldsymbol{X}} \,\Big|\, \boldsymbol{Y}; \boldsymbol{\theta}^{\text{old}}\Big) p(\,\boldsymbol{Y}).$$

6.4.1 The Gaussian mixture model and K-means

We return once again to the GMM, but this time, armed with the EM algorithm, we shall finally derive the learning rules. The complete cross entropy for a Gaussian mixture model is

$$\begin{aligned}
\mathrm{H}\big[\check{\boldsymbol{X}},\, \boldsymbol{Y}; \boldsymbol{\theta}\big] &\approx \big\langle -\log \hat{p}\big(\check{\boldsymbol{X}},\, \boldsymbol{Y}; \boldsymbol{\theta}\big)\big\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= \big\langle -\log \hat{p}\big(\,\boldsymbol{Y}\,\big|\,\check{\boldsymbol{X}}; \boldsymbol{\theta}\big)\hat{p}\big(\check{\boldsymbol{X}}; \boldsymbol{\theta}\big)\big\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= \left\langle -\log \prod_{k=1}^{K} \Big[\mathcal{N}\Big(\,\boldsymbol{Y}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\Big)\pi_k\Big]^{\check{X}_k} \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= \left\langle -\sum_{k=1}^{K} \check{X}_k \left[\frac{1}{2}\log\Big|{\boldsymbol{\Sigma}_k}^{-1}\Big| - \frac{1}{2}\Big(\boldsymbol{\mu}_k - \,\boldsymbol{Y}\Big){\boldsymbol{\Sigma}_k}^{-1}\Big(\boldsymbol{\mu}_k - \,\boldsymbol{Y}\Big) + \log \pi_k\right] \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} + C.
\end{aligned}$$

To enforce the fact that the source probabilities sum to one, we can augment the loss with a Lagrangian term:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right) + \mathrm{H}\big[\check{\boldsymbol{X}},\, \boldsymbol{Y}; \boldsymbol{\theta}\big] \\ &= \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right) + \left\langle -\sum_{k=1}^{K}\check{X}_k\left[\frac{1}{2}\log\left|{\boldsymbol{\Sigma}_k}^{-1}\right| - \frac{1}{2}\Big(\boldsymbol{\mu}_k - \boldsymbol{Y}\Big){\boldsymbol{\Sigma}_k}^{-1}\Big(\boldsymbol{\mu}_k - \boldsymbol{Y}\Big) + \log\pi_k\right]\right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} + C.\end{aligned}$$

We take the derivatives in turn. First the mixing proportions:

$$0 \overset{\text{set}}{=} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\pi_k} = \lambda - \frac{\langle\check{X}_k\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}}{\pi_k} \implies \sum_{k=1}^{K}\pi_k\lambda = \sum_{k=1}^{K}\langle\check{X}_k\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \implies \lambda = 1 \implies \pi_k = \langle\check{X}_k\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}};$$

then the class-conditional means:

$$0 \overset{\text{set}}{=} \frac{\partial\mathcal{L}}{\partial{\boldsymbol{\mu}_k}^{\mathrm{T}}} = \left\langle\check{X}_k\Big(\boldsymbol{\mu}_k - \boldsymbol{Y}\Big)^{\mathrm{T}}{\boldsymbol{\Sigma}_k}^{-1}\right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \implies \boldsymbol{\mu}_k = \frac{\langle\check{X}_k\,\boldsymbol{Y}\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}}{\langle\check{X}_k\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}};$$

and the class-conditional covariances:

$$\begin{aligned}0 \overset{\text{set}}{=} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}{\boldsymbol{\Sigma}_k}^{-1}} = \left\langle-\check{X}_k\left[\frac{1}{2}\boldsymbol{\Sigma}_k - \frac{1}{2}\Big(\boldsymbol{\mu}_k - \boldsymbol{Y}\Big)\Big(\boldsymbol{\mu}_k - \boldsymbol{Y}\Big)^{\mathrm{T}}\right]\right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \implies \boldsymbol{\Sigma}_k &= \frac{\left\langle\check{X}_k\Big(\boldsymbol{\mu}_k - \boldsymbol{Y}\Big)\Big(\boldsymbol{\mu}_k - \boldsymbol{Y}\Big)^{\mathrm{T}}\right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}}{\langle\check{X}_k\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}} \\ &= \frac{\langle\check{X}_k\,\boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}}{\langle\check{X}_k\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}} - \boldsymbol{\mu}_k{\boldsymbol{\mu}_k}^{\mathrm{T}}.\end{aligned}$$

Whether in EM or under a fully observed model, the optimal parameters are intuitive. The optimal mixing proportion, emission mean, and emission covariance for class k are their sample counterparts, i.e. the sample proportion, sample mean, and sample covariance (resp.); or, to put it yet another way, the average number of occurrences of class k, the average value of the samples $\boldsymbol{Y}$ from class k, and the average covariance of the samples $\boldsymbol{Y}$ from class k. The difference between learning (in EM) with latent, rather than fully observed, classes is that these are *weighted*, rather than unweighted, averages. In particular, class assignments are soft: each class takes some continuous-valued *responsibility* for each datum $\boldsymbol{y}_n$, namely $\mathbb{E}_{\hat{p}}\big[\check{\boldsymbol{X}}\big|\boldsymbol{y}_n\big] = \check{p}\big(\check{\boldsymbol{X}}\big|\boldsymbol{y}_n\big)$, the probability of that class under the (previous) recognition distribution.

class responsibilities

For example, when the class labels are observed, the denominator in the equation for the optimal mean becomes just the number of times class k occurred, and the numerator picks out just those samples $\boldsymbol{y}_n$ associated with class k. This is the sample average of the $\boldsymbol{y}_n$ in class k. But when the class $\check{\boldsymbol{X}}$ is latent, the denominator is the average *soft* assignment or responsibility of class k, and the numerator is the (soft) average of *all* observations $\boldsymbol{y}_n$, each weighted by the responsibility that class k takes for it.

Evidently, the expected sufficient statistics are $\langle\check{X}_k\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}$, $\langle\check{X}_k\,\boldsymbol{Y}\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}$, and $\langle\check{X}_k\,\boldsymbol{Y}\,\boldsymbol{Y}^{\mathrm{T}}\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}$. During EM, i.e. when the averaging distribution is....

K-means

In Section 2.1.1, we saw what happens to the recognition distribution of the GMM when all classes use the same covariance matrix, $\boldsymbol{\Sigma}$. The class boundaries become lines, and the responsibility of class j for observation $\boldsymbol{y}$ becomes

$$\hat{p}\Big(\hat{X}_j = 1\Big|\boldsymbol{y};\boldsymbol{\theta}\Big) = \frac{\exp\left\{-\frac{1}{2}\Big(\boldsymbol{y}-\boldsymbol{\mu}_j\Big)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\Big(\boldsymbol{y}-\boldsymbol{\mu}_j\Big)\right\}\pi_j}{\sum_{k=1}^{K}\exp\left\{-\frac{1}{2}\Big(\boldsymbol{y}-\boldsymbol{\mu}_k\Big)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\Big(\boldsymbol{y}-\boldsymbol{\mu}_k\Big)\right\}\pi_k}. \tag{6.7}$$

It is not hard to see that in the limit of infinite precision, this quantity goes to zero unless $\boldsymbol{y}$ is closer to $\boldsymbol{\mu}_j$ than any other mean, in which case it goes to 1. The "source" probabilities $\boldsymbol{\pi}$ have become irrelevant. Then the algorithm becomes

K-Means

- **E step**: $\check{p}_{t+1}\Big(\hat{X}_j = 1\Big|\boldsymbol{y}\Big) \overset{\text{set}}{=} \begin{cases} 1, & \text{if } j = \operatorname{argmin}_k \Big\|\boldsymbol{y}-\boldsymbol{\mu}_k\Big\| \\ 0, & \text{otherwise} \end{cases}$
- **M step**: $\boldsymbol{\mu}_j \overset{\text{set}}{=} \frac{\big\langle \check{p}_{t+1}\big(\hat{X}_j=1\big|\boldsymbol{Y}\big)\boldsymbol{Y}\big\rangle_{\boldsymbol{Y}}}{\big\langle \check{p}_{t+1}\big(\hat{X}_j=1\big|\boldsymbol{Y}\big)\big\rangle_{\boldsymbol{Y}}}$

This algorithm, which pre-existed EM for the GMM, is known as *K-means*. *K-means*

6.4.2 Hidden Markov model

The average is across all sequences.... Here we use *superscripts* for the "time variable" (t), i.e. to denote a random variable's order in the sequence, so that subscripts can continue to pick out a single entry (k) in the random vector $\check{\boldsymbol{X}}$. Bare random variables, i.e. without super- or subscripts, denote the complete set of observations. The elements of the state-transition matrix $\mathbf{A}$ are referred to as a_{ij}, with i and j indicating row and column, respectively. We also use the abbreviation $\hat{p}\Big(\hat{\boldsymbol{X}}^1\Big|\hat{\boldsymbol{X}}^0;\boldsymbol{\theta}\Big) := \hat{p}\Big(\hat{\boldsymbol{X}}^1;\boldsymbol{\theta}\Big) = \mathrm{Categ}(\boldsymbol{\pi})$.

$$\begin{aligned}
\mathrm{H}\big[\check{\boldsymbol{X}},\boldsymbol{Y};\boldsymbol{\theta}\big] &\approx \big\langle -\log\hat{p}\big(\check{\boldsymbol{X}},\boldsymbol{Y};\boldsymbol{\theta}\big)\big\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \\
&= \left\langle -\log\prod_{t=1}^{T}\hat{p}\Big(\check{\boldsymbol{X}}^t\Big|\check{\boldsymbol{X}}^{t-1};\boldsymbol{\theta}\Big)\hat{p}\Big(\boldsymbol{Y}^t\Big|\check{\boldsymbol{X}}^{t-1};\boldsymbol{\theta}\Big)\right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \\
&= \left\langle -\log\left(\prod_{t=2}^{T}\prod_{i=1}^{K}\prod_{j=1}^{K}a_{ij}^{\check{X}_i^t\check{X}_j^{t-1}}\right)\left(\prod_{k=1}^{K}\pi_k^{\check{X}_k^1}\right)\left(\prod_{t=1}^{T}\prod_{k=1}^{K}\mathcal{N}\Big(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\Big)^{\check{X}_k^t}\right)\right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \\
&= \left\langle -\sum_{t=2}^{T}\sum_{i=1}^{K}\sum_{j=1}^{K}\check{X}_i^t\check{X}_j^{t-1}\log a_{ij} - \sum_{k=1}^{K}\check{X}_k^1\log\pi_k - \sum_{t=1}^{T}\sum_{k=1}^{K}\check{X}_k^t\mathcal{N}\Big(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\Big)\right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}.
\end{aligned}$$

Again we need to enforce probabilities summing to one, in this case both the source probability $\boldsymbol{\pi}$ and all K columns of the state-transition matrix $\mathbf{A}$. The Lagrangian becomes

$$\mathcal{L}(\boldsymbol{\theta}) = \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right) + \sum_{j=1}^{K}\eta_j\left(\sum_{i=1}^{K}a_{ij} - 1\right) + \mathrm{H}\big[\check{\boldsymbol{X}},\boldsymbol{Y};\boldsymbol{\theta}\big].$$

The derivative with respect to π_k is exactly as above, just confined to the very first sample ($t = 1$), and so the optimal source probability is

$$\pi_k = \left\langle \check{X}_k^1 \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}.$$

Likewise for the class-conditional means and covariances, although here we have to remember to sum across all samples:

$$\boldsymbol{\mu}_k = \frac{\left\langle \sum_{t=1}^T \check{X}_k^t \, \boldsymbol{Y}^t \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}}{\left\langle \sum_{t=1}^T \check{X}_k^t \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}}, \qquad \boldsymbol{\Sigma}_k = \frac{\left\langle \sum_{t=1}^T \check{X}_k^t \, \boldsymbol{Y}^t \, \boldsymbol{Y}^{t\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}}{\left\langle \sum_{t=1}^T \check{X}_k^t \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}} - \boldsymbol{\mu}_k \boldsymbol{\mu}_k{}^{\mathrm{T}}.$$

Finally, we derive the optimal state-transition probabilities:

$$\begin{aligned}
0 \overset{\text{set}}{=} \frac{\partial \mathcal{L}}{\partial a_{ij}{}^{\mathrm{T}}} = \eta_j - \frac{\left\langle \sum_{t=2}^T \check{X}_t^i \check{X}_{t-1}^j \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}}{a_{ij}} \quad &\Longrightarrow \quad a_{ij} \eta_j = \left\langle \sum_{t=2}^T \check{X}_t^i \check{X}_{t-1}^j \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
&\Longrightarrow \quad \sum_{i=1}^K a_{ij} \eta_j = \left\langle \sum_{t=2}^T \sum_{i=1}^K \check{X}_t^i \check{X}_{t-1}^j \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
&\Longrightarrow \quad \eta_j = \left\langle \sum_{t=2}^T \check{X}_{t-1}^j \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
&\Longrightarrow \quad a_{ij} = \frac{\left\langle \sum_{t=2}^T \check{X}_t^i \check{X}_{t-1}^j \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}}{\left\langle \sum_{t=2}^T \check{X}_{t-1}^j \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}}
\end{aligned}$$

The optimal parameters again have intuitive interpretations. The class-conditional means and covariance are computed exactly as with the GMM, except that the averages are now across time and independent sequences rather than independent samples. In the case of EM, there is one other crucial distinction from the GMM: The recognition means of the HMM are conditioned on samples from *all time*; that is, they are the *smoother means*. This distinction has no relevance in the GMM, which has no temporal dimension.

Similarly, the optimal mixing proportions for the *initial* state look like the optimal mixing proportions for the latent classes in the GMM, although the sample average for the HMM is over sequences, rather than individual samples. One upshot is that it would be impossible to estimate the initial mixing proportions properly without access to multiple, independent sequences (this makes sense). And again, we must be careful in the case of EM: the expectation should be taken under the *smoother distribution over the initial state*. That is, to estimate the initial mixing proportions, the inference algorithm should first run all the way to the end (filter) and back (smoother)! This may at first be surprising, but note that future observations should indeed have some (albeit diminishing) influence on our belief about initial state. (We can imagine, colorfully, an unexpected future observation in light of which we revise our belief about the initial state: "Oh, I guess it must have started in state five, then....")

We turn to the remaining parameters, the elements of the state-transition matrix, $\mathbf{A}$. When the states have been fully observed, the optimal a_{ij} is merely the proportion of times state j transitioned to state i (rather than some other state). Under EM, the numerator

is the *expected* frequency of state j followed by i, where the expectation is taken under the smoother. To transform this into a transition probability, the base frequency of state j must be taken into account; under EM, we estimate it with its expected frequency under the smoother (averaged over sequences).

6.4.3 Factor analysis and principal-components analysis

Retaining the Gaussian emissions from the GMM but exchanging the categorical latent variable for a standard normal variate yields "factor analysis" (see Section 2.1.2). We also restrict the emission covariance to be diagonal in order to remove a degree of freedom that is not (as we shall see) identifiable from the data. The model is fully described by Eq. 2.20. However, we depart slightly from that formulation by augmenting the latent variable vector $\check{\boldsymbol{X}}$ with a "random" scalar that is 1 with probability 1. This allows us to absorb the bias $\boldsymbol{c}$ into a column of the emission matrix $\mathbf{C}$, reducing clutter without reducing generality.

The learning problem starts once again with minimization of the joint cross entropy:

$$\begin{aligned}
\mathrm{H}\big[\check{\boldsymbol{X}}, \boldsymbol{Y}; \boldsymbol{\theta}\big] &\approx \big\langle -\log \hat{p}\big(\check{\boldsymbol{X}}, \boldsymbol{Y}; \boldsymbol{\theta}\big)\big\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= \big\langle -\log \hat{p}\big(\boldsymbol{Y} \big| \check{\boldsymbol{X}}; \boldsymbol{\theta}\big)\hat{p}\big(\check{\boldsymbol{X}}; \boldsymbol{\theta}\big)\big\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= \big\langle -\log \mathcal{N}\big(\mathbf{C}\check{\boldsymbol{X}}, \mathbf{D}\big) - \log \mathcal{N}(\mathbf{0}, \mathbf{I})\big\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}
\end{aligned} \tag{6.8}$$

The model source distribution does not depend on any parameters, so only the model emission is differentiated. Starting with the emission matrix $\mathbf{C}$:

$$\begin{aligned}
0 \overset{\text{set}}{=} \frac{\mathrm{dH}}{\mathrm{d}\mathbf{C}} &= \left\langle -\frac{\mathrm{d}}{\mathrm{d}\mathbf{C}} \log \mathcal{N}\big(\mathbf{C}\check{\boldsymbol{X}}, \mathbf{D}\big) \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= \left\langle \frac{1}{2} \frac{\mathrm{d}}{\mathrm{d}\mathbf{C}} \big(\boldsymbol{Y} - \mathbf{C}\check{\boldsymbol{X}}\big)^{\mathrm{T}} \mathbf{D}^{-1} \big(\boldsymbol{Y} - \mathbf{C}\check{\boldsymbol{X}}\big) \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= \left\langle \mathbf{D}^{-1} \big(\boldsymbol{Y} - \mathbf{C}\check{\boldsymbol{X}}\big) \check{\boldsymbol{X}}^{\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
\implies \mathbf{C} &= \left\langle \boldsymbol{Y} \check{\boldsymbol{X}}^{\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \left\langle \check{\boldsymbol{X}} \check{\boldsymbol{X}}^{\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}^{-1},
\end{aligned}$$

the normal equations. Thus, in a fully observed model, finding $\mathbf{C}$ amounts to linear regression.

The emission covariance also takes on a familiar form:

$$\begin{aligned}
0 \overset{\text{set}}{=} \frac{\mathrm{dH}}{\mathrm{d}\mathbf{D}^{-1}} &= \left\langle -\frac{\mathrm{d}}{\mathrm{d}\mathbf{D}^{-1}} \log \mathcal{N}\big(\mathbf{C}\check{\boldsymbol{X}}, \mathbf{D}\big) \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= \left\langle -\frac{\mathrm{d}}{\mathrm{d}\mathbf{D}^{-1}} \left[\frac{1}{2} \log \big|\mathbf{D}^{-1}\big| - \frac{1}{2} \big(\boldsymbol{Y} - \mathbf{C}\check{\boldsymbol{X}}\big)^{\mathrm{T}} \mathbf{D}^{-1} \big(\boldsymbol{Y} - \mathbf{C}\check{\boldsymbol{X}}\big) \right] \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= \left\langle \mathbf{D} - \big(\boldsymbol{Y} - \mathbf{C}\check{\boldsymbol{X}}\big)\big(\boldsymbol{Y} - \mathbf{C}\check{\boldsymbol{X}}\big)^{\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} \\
\implies \mathbf{D} &= \left\langle \big(\boldsymbol{Y} - \mathbf{C}\check{\boldsymbol{X}}\big)\big(\boldsymbol{Y} - \mathbf{C}\check{\boldsymbol{X}}\big)^{\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}
\end{aligned}$$

The final line can be simplified using our newly acquired formula for $\mathbf{C}$. First expanding

the quadratic and then applying the identity:

$$\begin{aligned}\mathbf{D} &= \Big\langle \boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} - \boldsymbol{Y}\check{\boldsymbol{X}}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}} - \mathbf{C}\check{\boldsymbol{X}}\boldsymbol{Y}^{\mathrm{T}} - \mathbf{C}\check{\boldsymbol{X}}\check{\boldsymbol{X}}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}}\Big\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \\ &= \big\langle \boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}} - \Big\langle \boldsymbol{Y}\check{\boldsymbol{X}}^{\mathrm{T}}\Big\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}\mathbf{C}^{\mathrm{T}} - \mathbf{C}\big\langle\check{\boldsymbol{X}}\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} + \mathbf{C}\Big\langle\check{\boldsymbol{X}}\check{\boldsymbol{X}}^{\mathrm{T}}\Big\rangle_{\check{\boldsymbol{X}}}\mathbf{C}^{\mathrm{T}} \\ &= \big\langle \boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}} - \mathbf{C}\big\langle\check{\boldsymbol{X}}\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}.\end{aligned}$$

Now, we require $\mathbf{D}$ to be diagonal. It may be observed that the derivative with respect to any particular entry of $\mathbf{D}$ is independent of all other entries, so simply setting some components to zero does not change the optimum for the other components. (This can be made rigorous, albeit pedantic, with a Lagrangian formulation.) So we merely extract the diagonal from the final equation.

The E step. In Section 2.1.2, we derived the recognition distribution for factor analysis:

$$\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\boldsymbol{y};\boldsymbol{\theta}\Big) = \mathcal{N}\Big(\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\mathbf{C}^{\mathrm{T}}\mathbf{D}^{-1}\boldsymbol{y}, \big(\mathbf{C}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{C}+\mathbf{I}\big)^{-1}\Big). \tag{6.9}$$

In the E step, then, the expected sufficient statistics for $\mathbf{C}$ and $\mathbf{D}$ are therefore calculated as

$$\Big\langle \boldsymbol{Y}\mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{Y}}\Big[\check{\boldsymbol{X}}^{\mathrm{T}}\Big|\boldsymbol{Y}\Big]\Big\rangle_{\boldsymbol{Y}} = \Big\langle \boldsymbol{Y}\boldsymbol{Y}\mathbf{D}^{-1}\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\Big\rangle_{\boldsymbol{Y}} = \big\langle \boldsymbol{Y}\boldsymbol{Y}\big\rangle_{\boldsymbol{Y}}\mathbf{D}^{-1}\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}$$

and

$$\begin{aligned}\Big\langle\mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{Y}}\Big[\check{\boldsymbol{X}}\check{\boldsymbol{X}}^{\mathrm{T}}\Big|\boldsymbol{Y}\Big]\Big\rangle_{\boldsymbol{Y}} &= \Big\langle\mathrm{Cov}_{\check{\boldsymbol{X}}|\boldsymbol{Y}}\big[\check{\boldsymbol{X}}\big|\boldsymbol{Y}\big] + \mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{Y}}\big[\check{\boldsymbol{X}}\big|\boldsymbol{Y}\big]\mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{Y}}\Big[\check{\boldsymbol{X}}^{\mathrm{T}}\Big|\boldsymbol{Y}\Big]\Big\rangle_{\boldsymbol{Y}} \\ &= \Big\langle\big(\mathbf{C}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{C}+\mathbf{I}\big)^{-1} + \boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\mathbf{C}^{\mathrm{T}}\mathbf{D}^{-1}\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\Big\rangle_{\boldsymbol{Y}} \\ &= \big(\mathbf{C}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{C}+\mathbf{I}\big)^{-1} + \boldsymbol{\Sigma}_{\hat{x}|\hat{y}}\mathbf{C}^{\mathrm{T}}\mathbf{D}^{-1}\big\langle\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}}\mathbf{D}^{-1}\mathbf{C}\boldsymbol{\Sigma}_{\hat{x}|\hat{y}}.\end{aligned}$$

Principal-components analysis

We saw that in the limit of equal and infinite emission precisions, EM for the GMM reduces to K-means. Now we investigate this limit in the case of EM for the factor analyzer. In this case the only parameter to estimate is $\mathbf{C}$.

From Eq. 6.9, we see that the recognition covariance goes to zero as $\mathbf{D}^{-1}$ goes to infinity—recognition becomes deterministic. With slightly more work, we can also determine the mean, $\bar{\boldsymbol{x}}$, to which each $\boldsymbol{y}$ is deterministic assigned. Setting $\mathbf{D} = \epsilon\mathbf{I}$, we find

$$\bar{\boldsymbol{X}} := \mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{Y}}\Big[\check{\boldsymbol{X}}^{\mathrm{T}}\Big|\boldsymbol{Y}\Big] = \Big(\mathbf{C}^{\mathrm{T}}\frac{\mathbf{I}}{\epsilon}\mathbf{C}+\mathbf{I}\Big)^{-1}\mathbf{C}^{\mathrm{T}}\frac{\mathbf{I}}{\epsilon}\boldsymbol{Y} = \big(\mathbf{C}^{\mathrm{T}}\mathbf{C}+\epsilon\mathbf{I}\big)^{-1}\mathbf{C}^{\mathrm{T}}\boldsymbol{Y} \xrightarrow[\epsilon\to 0]{} \big(\mathbf{C}^{\mathrm{T}}\mathbf{C}\big)^{-1}\mathbf{C}^{\mathrm{T}}\boldsymbol{Y}.$$

The final expression is the Moore-Penrose pseudo-inverse of $\mathbf{C}$, i.e., the latent-space projection of $\boldsymbol{Y}$ that yields the smallest reconstruction error under the emission matrix $\mathbf{C}$.

[.....]

[**?**]

Iterative Principal-Components Analysis

- **E step**: $\bar{\boldsymbol{X}} \overset{\text{set}}{=} \big(\mathbf{C}^{\mathrm{T}}\mathbf{C}\big)^{-1}\mathbf{C}^{\mathrm{T}}\boldsymbol{Y}$
- **M step**: $\mathbf{C}\overset{\text{set}}{=}\Big\langle\boldsymbol{Y}\bar{\boldsymbol{X}}^{\mathrm{T}}\Big\rangle_{\boldsymbol{Y}}\Big\langle\bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^{\mathrm{T}}\Big\rangle_{\boldsymbol{Y}}^{-1}$,

6.4.4 Linear-Gaussian state-space models

Exactly analogously to the extension of GMMs to HMMs, we extend factor analysis through time (or space) into a dynamical system. Now the state is a vector of continuous values, and assumed to be normally distributed about a linear function of its predecessor.....

We again forebear to specify the averaging distribution....

$$\begin{aligned}
\mathrm{H}\big[\check{\boldsymbol{X}},\, \boldsymbol{Y};\boldsymbol{\theta}\big] &\approx \big\langle -\log \hat{p}\big(\check{\boldsymbol{X}},\, \boldsymbol{Y};\boldsymbol{\theta}\big)\big\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \\
&= \left\langle -\log \prod_{t=1}^{T} \hat{p}\Big(\check{\boldsymbol{X}}^t\Big|\check{\boldsymbol{X}};\boldsymbol{\theta}\Big)\hat{p}\big(\boldsymbol{Y}\big|\check{\boldsymbol{X}};\boldsymbol{\theta}\big)\right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \\
&= \left\langle -\sum_{t=2}^{T} \log \mathcal{N}\Big(\mathbf{A}\check{\boldsymbol{X}}^{t-1}, \boldsymbol{\Sigma}_{\hat{x}}\Big) - \sum_{t=1}^{T} \log \mathcal{N}\Big(\mathbf{C}\check{\boldsymbol{X}}^{t}, \boldsymbol{\Sigma}_{\hat{y}|\hat{x}}\Big) - \log \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)\right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}.
\end{aligned}$$

All three summands are Gaussians, so we only consider in detail the differentiation of one. Optimizing first with respect to $\mathbf{A}$, we find again the familiar normal equations:

$$\begin{aligned}
0 \overset{\text{set}}{=} \frac{\mathrm{dH}}{\mathrm{d}\mathbf{A}} &= \left\langle -\sum_{t=2}^{T} \frac{\mathrm{d}}{\mathrm{d}\mathbf{A}} \log \mathcal{N}\Big(\mathbf{A}\check{\boldsymbol{X}}^{t-1}, \boldsymbol{\Sigma}_{\hat{x}}\Big)\right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \\
&= \left\langle \sum_{t=2}^{T} \frac{\mathrm{d}}{\mathrm{d}\mathbf{A}} \frac{1}{2} (\check{\boldsymbol{X}}^t - \mathbf{A}\check{\boldsymbol{X}}^{t-1})^{\mathrm{T}} \boldsymbol{\Sigma}_{\hat{x}}^{-1} (\check{\boldsymbol{X}}^t - \mathbf{A}\check{\boldsymbol{X}}^{t-1})\right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \\
&= \left\langle \sum_{t=2}^{T} \boldsymbol{\Sigma}_{\hat{x}}^{-1} (\check{\boldsymbol{X}}^t - \mathbf{A}\check{\boldsymbol{X}}^{t-1}) \check{\boldsymbol{X}}^{t-1}\right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \\
\implies \mathbf{A} &= \left(\sum_{t=2}^{T} \Big\langle \check{\boldsymbol{X}}^t \check{\boldsymbol{X}}^{t-1\mathrm{T}}\Big\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}\right)\left(\sum_{t=2}^{T} \Big\langle \check{\boldsymbol{X}}^{t-1} \check{\boldsymbol{X}}^{t-1\mathrm{T}}\Big\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}\right)^{-1} \\
&= \Big\langle \check{\boldsymbol{X}}^t \check{\boldsymbol{X}}^{t-1\mathrm{T}}\Big\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \Big\langle \check{\boldsymbol{X}}^{t-1} \check{\boldsymbol{X}}^{t-1\mathrm{T}}\Big\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}^{-1},
\end{aligned}$$

where in the last line we intepret the average to be under samples from within, as well as across, sequences. Turning to the covariance matrix,

IMPLEMENTATION NOTE: **Since each sequence contributes only $T-1$ samples, one must remember to subtract $N_{\text{sequences}}$ from the total number of samples across sequences before normalizing.**

$$\begin{aligned}
0 \overset{\text{set}}{=} \frac{\mathrm{dH}}{\mathrm{d}\boldsymbol{\Sigma}_{\hat{x}}^{-1}} &= \left\langle -\sum_{t=2}^{T} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\Sigma}_{\hat{x}}^{-1}} \log \mathcal{N}\Big(\mathbf{A}\check{\boldsymbol{X}}^{t-1}, \boldsymbol{\Sigma}_{\hat{x}}\Big)\right\rangle_{\check{\boldsymbol{X}}} \\
&= \left\langle -\sum_{t=2}^{T} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\Sigma}_{\hat{x}}^{-1}} \left[\frac{1}{2}\log\big|\boldsymbol{\Sigma}_{\hat{x}}^{-1}\big| - \frac{1}{2}(\check{\boldsymbol{X}}^t - \mathbf{A}\check{\boldsymbol{X}}^{t-1})^{\mathrm{T}} \boldsymbol{\Sigma}_{\hat{x}}^{-1} (\check{\boldsymbol{X}}^t - \mathbf{A}\check{\boldsymbol{X}}^{t-1})\right]\right\rangle_{\check{\boldsymbol{X}}} \\
&= \left\langle -\sum_{t=2}^{T} \Big[\boldsymbol{\Sigma}_{\hat{x}} - (\check{\boldsymbol{X}}^t - \mathbf{A}\check{\boldsymbol{X}}^{t-1})(\check{\boldsymbol{X}}^t - \mathbf{A}\check{\boldsymbol{X}}^{t-1})^{\mathrm{T}}\Big]\right\rangle_{\check{\boldsymbol{X}}} \\
\implies \boldsymbol{\Sigma}_{\hat{x}} &= \frac{1}{T-1}\sum_{t=2}^{T} \Big\langle (\check{\boldsymbol{X}}^t - \mathbf{A}\check{\boldsymbol{X}}^{t-1})(\check{\boldsymbol{X}}^t - \mathbf{A}\check{\boldsymbol{X}}^{t-1})^{\mathrm{T}}\Big\rangle_{\check{\boldsymbol{X}}} \\
&= \Big\langle (\check{\boldsymbol{X}}^t - \mathbf{A}\check{\boldsymbol{X}}^{t-1})(\check{\boldsymbol{X}}^t - \mathbf{A}\check{\boldsymbol{X}}^{t-1})^{\mathrm{T}}\Big\rangle_{\check{\boldsymbol{X}}} \\
&= \Big\langle \check{\boldsymbol{X}}^t \check{\boldsymbol{X}}^{t\mathrm{T}}\Big\rangle_{\check{\boldsymbol{X}}} - \mathbf{A}\Big\langle \check{\boldsymbol{X}}^{t-1} \check{\boldsymbol{X}}^{t\mathrm{T}}\Big\rangle_{\check{\boldsymbol{X}}}.
\end{aligned}$$

where again in the penultimate line we changed the interpretation of the average to be within as well as across sequences. The final simplification was carried out with the equation for $\mathbf{A}$, exactly the same as with factor analysis (see above).

Derivations precisely analogous lead to formalae for the other cumulants. Here are all of the equations:

$$\begin{aligned}
\mathbf{A} &= \left\langle \check{\boldsymbol{X}}^t \check{\boldsymbol{X}}^{t-1\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \left\langle \check{\boldsymbol{X}}^{t-1} \check{\boldsymbol{X}}^{t-1\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}^{-1} & \boldsymbol{\Sigma}_{\hat{x}} &= \left\langle \check{\boldsymbol{X}}^t \check{\boldsymbol{X}}^{t\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}} - \mathbf{A} \left\langle \check{\boldsymbol{X}}^{t-1} \check{\boldsymbol{X}}^{t\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}} \\
\mathbf{C} &= \left\langle \boldsymbol{Y}^t \check{\boldsymbol{X}}^{t\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \left\langle \check{\boldsymbol{X}}^t \check{\boldsymbol{X}}^{t\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}}^{-1} & \boldsymbol{\Sigma}_{\hat{y}|\hat{x}} &= \left\langle \check{\boldsymbol{X}}^t \check{\boldsymbol{X}}^{t\mathrm{T}} \right\rangle_{\boldsymbol{Y}} - \mathbf{C} \left\langle \check{\boldsymbol{X}}^{t-1} \check{\boldsymbol{X}}^{t\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} \\
\boldsymbol{\mu}_1 &= \left\langle \check{\boldsymbol{X}}^1 \right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} & \boldsymbol{\Sigma}_1 &= \left\langle \check{\boldsymbol{X}}^t \check{\boldsymbol{X}}^{t\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} - \boldsymbol{\mu}_1 \boldsymbol{\mu}_1{}^{\mathrm{T}}.
\end{aligned} \tag{6.10}$$

The E step. In Eq. 6.10, the bracketed quantities (including the brackets) are the *sufficient statistics*. They are all averages of either vectors or outer products of vectors, reflecting the quadratic structure inherent in normal distributions. When the states $\check{\boldsymbol{X}}^t = \boldsymbol{X}^t$ are observed, these outer products are simply averaged under the data distribution. In the context of the EM algorithm, where the states are unobserved, the relevant averaging distributions $\check{p}$ are the RTS smoothing distributions, multiplied by the data distribution:

$$\begin{aligned}
\check{p}\Big(\check{\boldsymbol{X}}^t, \boldsymbol{Y}\Big) &= \hat{p}\Big(\check{\boldsymbol{X}}^t \Big| \boldsymbol{Y}; \theta^{\text{old}}\Big) p(\boldsymbol{Y}) \\
\check{p}\Big(\check{\boldsymbol{X}}^t, \check{\boldsymbol{X}}^{t-1}, \boldsymbol{Y}\Big) &= \hat{p}\Big(\check{\boldsymbol{X}}^t, \check{\boldsymbol{X}}^{t-1} \Big| \boldsymbol{Y}; \theta^{\text{old}}\Big) p(\boldsymbol{Y}).
\end{aligned}$$

(We emphasize again that $\boldsymbol{Y} := \boldsymbol{Y}^1, \ldots, \boldsymbol{Y}^T$.) In EM, Eq. 6.10 corresponds to the M step. Computing the sufficient statistics belongs to the E step. That is, the E step requires first running the RTS smoother—more specifically, the Kalman filter followed by the RTS smoother—and then computing the sufficient statistics under them. To make this extremely concrete, we note that having the smoother distribution in hand means having a mean (vector) and covariance matrix at every time step, since the distribution is normal. To compute expected outer products, then, we have to combine these together.

For $\mathbf{A}$ and $\boldsymbol{\Sigma}_{\hat{x}}$ we need

$$\begin{aligned}
\left\langle \check{\boldsymbol{X}}^{t-1} \check{\boldsymbol{X}}^{t-1\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}} &= \left\langle \frac{1}{T-1} \sum_{t=2}^{T} \Big[\mathrm{Cov}\Big[\check{\boldsymbol{X}}^{t-1} \Big| \boldsymbol{Y} \Big] + \mathbb{E}\Big[\check{\boldsymbol{X}}^{t-1} \Big| \boldsymbol{Y} \Big] \mathbb{E}\Big[\check{\boldsymbol{X}}^{t-1\mathrm{T}} \Big| \boldsymbol{Y} \Big] \Big] \right\rangle_{\boldsymbol{Y}} \\
\left\langle \check{\boldsymbol{X}}^t \check{\boldsymbol{X}}^{t-1\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}} &= \left\langle \frac{1}{T-1} \sum_{t=2}^{T} \Big[\mathrm{Cov}\Big[\check{\boldsymbol{X}}^t, \check{\boldsymbol{X}}^{t-1} \Big| \boldsymbol{Y} \Big] + \mathbb{E}\Big[\check{\boldsymbol{X}}^t \Big| \boldsymbol{Y} \Big] \mathbb{E}\Big[\check{\boldsymbol{X}}^{t-1\mathrm{T}} \Big| \boldsymbol{Y} \Big] \Big] \right\rangle_{\boldsymbol{Y}} \\
\left\langle \check{\boldsymbol{X}}^t \check{\boldsymbol{X}}^{t\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}} &= \left\langle \frac{1}{T-1} \sum_{t=2}^{T} \Big[\mathrm{Cov}\Big[\check{\boldsymbol{X}}^t \Big| \boldsymbol{Y} \Big] + \mathbb{E}\Big[\check{\boldsymbol{X}}^t \Big| \boldsymbol{Y} \Big] \mathbb{E}\Big[\check{\boldsymbol{X}}^{t\mathrm{T}} \Big| \boldsymbol{Y} \Big] \Big] \right\rangle_{\boldsymbol{Y}}.
\end{aligned}$$

Notice that all sums *start at the second index*, and (consequently) *are normalized by* $T-1$. That is why the first and third statistics are not identical. If the sequence is long enough, including all T terms in the first and last statistics will probably have little effect, but both averages can be computed with very little computational cost.

For $\mathbf{C}$ and $\boldsymbol{\Sigma}_{\hat{y}|\hat{x}}$, in contrast, we collect statistics for *all* time. Thus, even though we also need a statistic we have called $\left\langle \check{\boldsymbol{X}}^t \check{\boldsymbol{X}}^{t\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}}$, in this case the average is over all t:

$$\begin{aligned}
\left\langle \check{\boldsymbol{X}}^t \check{\boldsymbol{X}}^{t\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}} &= \left\langle \frac{1}{T} \sum_{t=1}^{T} \Big[\mathrm{Cov}\Big[\check{\boldsymbol{X}}^t \,|\, \boldsymbol{Y} \Big] + \mathbb{E}\Big[\check{\boldsymbol{X}}^t \,|\, \boldsymbol{Y} \Big] \mathbb{E}\Big[\check{\boldsymbol{X}}^{t\mathrm{T}} \,|\, \boldsymbol{Y} \Big] \Big] \right\rangle_{\boldsymbol{Y}} \\
\left\langle \boldsymbol{Y}^t \check{\boldsymbol{X}}^{t\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}},\boldsymbol{Y}} &= \left\langle \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{Y}^t \mathbb{E}\Big[\check{\boldsymbol{X}}^{t\mathrm{T}} \,|\, \boldsymbol{Y} \Big] \right\rangle_{\boldsymbol{Y}} \\
\left\langle \boldsymbol{Y}^t \boldsymbol{Y}^{t\mathrm{T}} \right\rangle_{\boldsymbol{Y}} &= \left\langle \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{Y}^t \boldsymbol{Y}^{t\mathrm{T}} \right\rangle_{\boldsymbol{Y}}
\end{aligned}$$

Finally, the sufficient statistics for the initial cumulants, $\boldsymbol{\mu}_1$ and $\boldsymbol{\Sigma}_1$, require no sums at all, since they rely only on the first time step:

$$\begin{aligned}
\left\langle \check{\boldsymbol{X}}^1 \right\rangle_{\check{\boldsymbol{X}}} &= \left\langle \mathbb{E}\Big[\check{\boldsymbol{X}}^1 \,|\, \boldsymbol{Y} \Big] \right\rangle_{\boldsymbol{Y}} \\
\left\langle \check{\boldsymbol{X}}^1 \check{\boldsymbol{X}}^{1\mathrm{T}} \right\rangle_{\check{\boldsymbol{X}}} &= \left\langle \mathbb{E}\Big[\check{\boldsymbol{X}}^1 \check{\boldsymbol{X}}^{1\mathrm{T}} \,|\, \boldsymbol{Y} \Big] \right\rangle_{\boldsymbol{Y}}
\end{aligned}$$

IMPLEMENTATION NOTE: **The second of these *can yield a low-rank covariance matrix if only a single trajectory has been observed*, and care is sometimes required to avoid inverting a singular matrix in the Kalman filter.**

6.5 Parameterized proxy recognition distributions

To widen our view of target models for EM, let us drop the very restrictive assumption that the model recognition distribution, $\hat{p}\Big(\hat{\boldsymbol{X}} \Big| \hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big)$, can be computed. It will therefore no longer be possible to tighten the bound completely in the E step. At this price, however, we have bought the ability to work with much more complicated generative models. The form of the proxy recognition distribution will now be fixed, not by the model recognition distribution, but by our choice.

6.5.1 Gaussian proxy recognition distributions and sparse coding

For continuous random latent variables, the simplest choice is perhaps the (multivariate) normal distribution:

$$\check{p}(\check{\boldsymbol{X}} \,|\, \boldsymbol{Y}) = \mathcal{N}(\boldsymbol{\nu}_{\boldsymbol{Y}}, \mathbf{P}_{\boldsymbol{Y}}^{-1}).$$

Note that we are allowing the mean and precision to depend on the observations, although we have not as yet specified how. To derive an EM algorithm under this proxy recognition distribution, we start with the free energy. Recall (from the first line of Eq. 6.6) that the free energy can be written as the difference of two entropies: the cross entropy of the joint distribution, and the entropy of the proxy recognition distribution (averaged under the data). The latter is now just the entropy of a multivariate normal distribution, averaged under the data:

$$\begin{aligned}
\mathbb{E}_{\check{\boldsymbol{X}},\boldsymbol{Y}}\big[-\log \check{p}(\check{\boldsymbol{X}} \,|\, \boldsymbol{Y}) \big] &= \mathbb{E}_{\check{\boldsymbol{X}},\boldsymbol{Y}}\left[-\frac{1}{2} \log|\mathbf{P}_{\boldsymbol{Y}}| + \frac{K}{2} \log \tau + \frac{1}{2} (\check{\boldsymbol{X}} - \boldsymbol{\nu}_{\boldsymbol{Y}})^{\mathrm{T}} \mathbf{P}_{\boldsymbol{Y}} (\check{\boldsymbol{X}} - \boldsymbol{\nu}_{\boldsymbol{Y}}) \right] \\
&= \frac{1}{2} (K \log \tau + K - \mathbb{E}_{\boldsymbol{Y}}[\log|\mathbf{P}_{\boldsymbol{Y}}|]).
\end{aligned} \tag{6.11}$$

Subtracting this from the cross entropy of the joint, we have

$$\mathcal{F}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\check{\boldsymbol{X}}, \boldsymbol{Y}}\big[-\log \hat{p}\big(\check{\boldsymbol{X}}, \boldsymbol{Y}; \boldsymbol{\theta}\big)\big] + \mathbb{E}_{\boldsymbol{Y}}\left[\frac{1}{2}\log|\mathbf{P}_{\boldsymbol{Y}}|\right] + C, \tag{6.12}$$

lumping constant terms into C. Notice that we have introduced the symbol $\boldsymbol{\phi}$ to stand for all the (currently unspecified) parameters of the proxy recognition distributions.

Local parameters

Now let us assume that we will learn (in the E step) one pair of parameters for each observation, $\boldsymbol{y}$; that is,

$$\boldsymbol{\phi} = \Big\{\boldsymbol{\nu}_{\boldsymbol{y}_1}, \mathbf{P}_{\boldsymbol{y}_1}, \ldots, \boldsymbol{\nu}_{\boldsymbol{y}_N}, \mathbf{P}_{\boldsymbol{y}_N}\Big\}. \tag{6.13}$$

The advantage of this approach is flexibility: the mean vector and precision matrix for each datum can take on arbitrary values. The (potential) disadvantage is inefficiency: Unless the parameters can be derived in closed-form, this approach will not scale well to a large number of observations. Indeed, the recognition distribution for a new (or held-out) observation $\boldsymbol{y}_{N+1}$ cannot be fit any faster, or predicted, on the basis of the previous observations. We shall explore alternatives shortly.

The E step. It is instructive [19] to begin the E-step optimization, even without having specified a generative model. Replacing the expectation over the data in Eq. 6.12 with a sample average and differentiating with respect to the recognition parameters $(\boldsymbol{\nu}_{\boldsymbol{y}}, \mathbf{P}_{\boldsymbol{y}})$ associated with a single datum $(\boldsymbol{y})$ clearly eliminates all the terms that depend on the parameters for other data. So we can drop the expectation over the data altogether. The gradient with respect to a recognition mean vector is then

$$\begin{aligned}
\frac{\partial \mathcal{F}}{\partial \boldsymbol{\nu}_{\boldsymbol{y}}} &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\nu}_{\boldsymbol{y}}}\mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{y}}\big[-\log \hat{p}\big(\check{\boldsymbol{X}}, \boldsymbol{y}; \boldsymbol{\theta}\big)\big|\boldsymbol{y}\big] \\
&= \mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{y}}\left[-\frac{\partial \log \hat{p}}{\partial \hat{\boldsymbol{x}}}\big(\check{\boldsymbol{X}}, \boldsymbol{y}; \boldsymbol{\theta}\big)\middle|\boldsymbol{y}\right] \\
&= \mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{y}}\left[\frac{\partial E_{\text{recog}}}{\partial \hat{\boldsymbol{x}}}\big(\check{\boldsymbol{X}}, \boldsymbol{y}, \boldsymbol{\theta}\big)\middle|\boldsymbol{y}\right] \overset{\text{set}}{=} 0.
\end{aligned} \tag{6.14}$$

The second line can be derived by exploiting the symmetry of $\boldsymbol{\nu}_{\boldsymbol{y}}$ and $\hat{\boldsymbol{X}}$ in a Gaussian function, and then integrating by parts. In the third line we have put the gradient in terms of the energy of the model recognition distribution, which can easily be seen to have the same gradient as the negative log of the model joint. Similarly, the gradient with respect to a recognition covariance matrix is

Exercise ??

$$\begin{aligned}
\frac{\partial \mathcal{F}}{\partial \mathbf{P}_{\boldsymbol{y}}^{-1}} &= \frac{\mathrm{d}}{\mathrm{d}\mathbf{P}_{\boldsymbol{y}}^{-1}}\left(\mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{y}}\big[-\log \hat{p}\big(\check{\boldsymbol{X}}, \boldsymbol{y}; \boldsymbol{\theta}\big)\big|\boldsymbol{y}\big] - \frac{1}{2}\log\left|\mathbf{P}_{\boldsymbol{y}}^{-1}\right|\right) \overset{\text{set}}{=} 0 \\
\implies \mathbf{P}_{\boldsymbol{y}} &= 2\frac{\mathrm{d}}{\mathrm{d}\mathbf{P}_{\boldsymbol{y}}^{-1}}\mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{y}}\big[-\log \hat{p}\big(\check{\boldsymbol{X}}, \boldsymbol{y}; \boldsymbol{\theta}\big)\big|\boldsymbol{y}\big] \\
&= \mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{y}}\left[-\frac{\partial^2 \log \hat{p}}{\partial \hat{\boldsymbol{x}} \partial \hat{\boldsymbol{x}}^{\mathrm{T}}}\big(\check{\boldsymbol{X}}, \boldsymbol{y}; \boldsymbol{\theta}\big)\middle|\boldsymbol{y}\right] \\
&= \mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{y}}\left[\frac{\partial^2 E_{\text{recog}}}{\partial \hat{\boldsymbol{x}} \partial \hat{\boldsymbol{x}}^{\mathrm{T}}}\big(\check{\boldsymbol{X}}, \boldsymbol{y}, \boldsymbol{\theta}\big)\middle|\boldsymbol{y}\right].
\end{aligned} \tag{6.15}$$

The third line can be derived by meticuously differentiating the normal distribution (or, more precisely, $f(\hat{\boldsymbol{x}}) \log f(\hat{\boldsymbol{x}})$ for normal $f(\hat{\boldsymbol{x}})$) with respect to both $\mathbf{P}_{\boldsymbol{y}}^{-1}$ and $\hat{\boldsymbol{x}}$ (twice), and then integrating by parts; or, more elegantly, by expressing the expectations in the Fourier domain (using Plancherel's theorem) and differentiating there [19]. EXERCISE ??

We have written the optimizations for the mean and precision in the form of Eqs. 6.14 and 6.15 in order to highlight the resemblance to the Laplace approximation encountered in Chapter 2 (see especially Eq. 2.32) [19]. In particular, Eqs. 6.14 and 6.15 become the Laplace approximation when the expectations commute with the energy-gradient and energy-Hessian functions of $\hat{\boldsymbol{x}}$. This happens precisely when the order of the energy in $\hat{\boldsymbol{x}}$ is quadratic or less.

Independent sources and Gaussian emissions: sparse coding and other models

To make the procedure more concrete, let us make a few further assumptions. Although this is by no means obligatory, the emissions have been Gaussian in all our examples heretofore, so we shall make the same choice here. For simplicity, we shall even let the noise be isotropic. The source random variables, on the other hand, we shall assume to be independent, but otherwise distributed generically:

$$\hat{p}\Big(\hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{k=1}^{K} \exp\Big\{-E_k(\hat{X}_k, \boldsymbol{\theta})\Big\}, \qquad \hat{p}\Big(\hat{\boldsymbol{Y}} \,\Big|\, \hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big) = \mathcal{N}\bigg(\mathbf{C}\hat{\boldsymbol{X}}, \frac{1}{\lambda}\mathbf{I}\bigg). \tag{6.16}$$

Among other models, Eq. 6.16 includes the *sparse-coding* models encountered in Section 2.1.3, in which each emission $\hat{\boldsymbol{y}}$ is assembled from a modest number of basis vectors drawn from a large (overcomplete) dictionary—i.e., an emission matrix $\mathbf{C}$ that is "fat." To enforce sparsity, the energy functions for the source variables can be chosen from leptokurtotic distributions. For example, we considered energy functions for the Laplace distribution and a smooth approximation thereof:

$$\begin{aligned} E_k(\hat{X}_k) &:= \alpha_k \Big|\hat{X}_k\Big| \\ &\approx \frac{\alpha_k}{\beta} \log\Big(\cosh(\beta \hat{X}_k)\Big) \quad \text{for large } \beta. \end{aligned} \tag{6.17}$$

This ensures that the source variables $\hat{X}_k$ take on values close to zero more frequently than normal random variables with similar variance. Unfortunately, as we saw, such fat-tailed distributions are not conjugate to the likelihood for the basis coefficients specified by Eq. 6.16, i.e. the Gaussian emission density, so they require approximate inference.

Here we are considering Gaussian proxy recognition distributions. For the sake of generality, however, we shall refrain at first from assuming any form for the energy functions, deriving our results for the more general class described by Eq. 6.16. Now, the cross entropy of this joint distribution is:

$$\begin{aligned} \mathbb{E}_{\check{\boldsymbol{X}}, \boldsymbol{Y}}\big[-\log \hat{p}\big(\check{\boldsymbol{X}}, \boldsymbol{Y}; \boldsymbol{\theta}\big)\big] &= \mathbb{E}_{\check{\boldsymbol{X}}, \boldsymbol{Y}}\left[\frac{K}{2}\log\lambda + \frac{\lambda}{2}\big\|\boldsymbol{Y} - \mathbf{C}\check{\boldsymbol{X}}\big\|^2 + \log Z(\boldsymbol{\theta}) + \sum_{k=1}^{K} E_k(\check{X}_k, \boldsymbol{\theta})\right] \\ &= \frac{K}{2}\log\lambda + \log Z(\boldsymbol{\theta}) + \frac{1}{2}\mathbb{E}_{\check{\boldsymbol{X}}, \boldsymbol{Y}}\left[\lambda\big\|\boldsymbol{Y} - \mathbf{C}\check{\boldsymbol{X}}\big\|^2 + 2\sum_{k=1}^{K} E_k(\check{X}_k, \boldsymbol{\theta})\right]. \end{aligned} \tag{6.18}$$

So far this follows directly from Eq. 6.16. The third term can be simplified further, however, by applying our assumption that the proxy recognition distribution is normal, and using Eq. A.17 from the appendix:

$$\mathbb{E}_{\check{\boldsymbol{X}},\boldsymbol{Y}}\Big[\lambda\big\|\,\boldsymbol{Y}-\mathbf{C}\check{\boldsymbol{X}}\,\big\|^2\Big]=\lambda\mathbb{E}_{\boldsymbol{Y}}\Big[\mathrm{tr}\big[\mathbf{C}\boldsymbol{\mathrm{P}}_{\boldsymbol{Y}}^{-1}\mathbf{C}^{\mathrm{T}}\big]+\|\,\boldsymbol{Y}-\mathbf{C}\boldsymbol{\nu}_{\boldsymbol{Y}}\|^2\Big].$$

The M step. We can now write a more specific expression for the free energy that will allow us to see how to carry out the M step, as well. Let us simplify even further and treat λ and the parameters of the prior as fixed. Then substituting the cross entropy of the joint, Eq. 6.18, into the free energy for Gaussian proxy recognition distributions, Eq. 6.12, yields

$$\mathcal{F}(\boldsymbol{\theta},\boldsymbol{\phi})=\mathbb{E}_{\boldsymbol{Y}}\left[\frac{\lambda}{2}\mathrm{tr}\big[\mathbf{C}\boldsymbol{\mathrm{P}}_{\boldsymbol{Y}}^{-1}\mathbf{C}^{\mathrm{T}}\big]+\frac{\lambda}{2}\|\,\boldsymbol{Y}-\mathbf{C}\boldsymbol{\nu}_{\boldsymbol{Y}}\|^2+\frac{1}{2}\log|\boldsymbol{\mathrm{P}}_{\boldsymbol{Y}}|\right]+\sum_{k=1}^{K}\mathbb{E}_{\check{\boldsymbol{X}},\boldsymbol{Y}}\big[E_k(\check{X}_k,\boldsymbol{\theta})\big]+C. \tag{6.19}$$

(All "constant" terms have been lumped into a single scalar C.) The only remaining model parameter to be optimized is the emission matrix, $\mathbf{C}$. We can compute the gradient of the free energy, Eq. 6.19, with respect to $\mathbf{C}$ using some results from the appendix (Section A.1):

$$\begin{aligned}\frac{\partial\mathcal{F}}{\partial\mathbf{C}}&=\lambda\mathbb{E}_{\boldsymbol{Y}}\big[\mathbf{C}\boldsymbol{\mathrm{P}}_{\boldsymbol{Y}}^{-1}-(\,\boldsymbol{Y}-\mathbf{C}\boldsymbol{\nu}_{\boldsymbol{Y}})\boldsymbol{\nu}_{\boldsymbol{Y}}{}^{\mathrm{T}}\big]\overset{\text{set}}{=}0\\ \implies\mathbf{C}&=\big\langle\,\boldsymbol{Y}\boldsymbol{\nu}_{\boldsymbol{Y}}{}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}}\big\langle\boldsymbol{\mathrm{P}}_{\boldsymbol{Y}}^{-1}+\boldsymbol{\nu}_{\boldsymbol{Y}}\boldsymbol{\nu}_{\boldsymbol{Y}}{}^{\mathrm{T}}\big\rangle_{\boldsymbol{Y}}^{-1},\end{aligned} \tag{6.20}$$

Thus, the emission matrix is once again given by some version of the normal equations. In practice, however, if each $\boldsymbol{y}$ is (e.g.) an image, and the model is overcomplete, then the matrix inversions will be computationally challenging. Instead, the gradient in Eq. 6.20 (or the natural gradient [14, 16]) can be descended.

The E step. To carry out the E step, we apply Eqs. 6.14 and 6.15 to the energy of the model recognition or joint distributions (they are equivalent up to constant terms), to wit, the bracketed terms in Eq. 6.18. Let us also commit to the source energies specified by Eq. 6.17, i.e., the energy of the Laplace distribution. Then the resulting recognition energy,

$$E_{\mathrm{recog}}(\hat{\boldsymbol{x}},\boldsymbol{y},\boldsymbol{\theta})=\frac{\lambda}{2}\|\,\boldsymbol{Y}-\mathbf{C}\hat{\boldsymbol{x}}\|^2+\sum_{k=1}^{K}\alpha_k\Big|\hat{x}_k\Big|,$$

is quadratic in $\hat{\boldsymbol{x}}$ everywhere except at points where $\hat{x}_k$ is zero (for any k). Therefore the energy-gradient function (of $\hat{\boldsymbol{x}}$) and the expectation in Eq. 6.14 "almost" commute with each other, and the free-energy gradient becomes

$$\begin{aligned}\frac{\partial\mathcal{F}}{\partial\boldsymbol{\nu}_{\boldsymbol{y}}}&=\mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{y}}\left[\frac{\partial E_{\mathrm{recog}}}{\partial\hat{\boldsymbol{x}}}\big(\check{\boldsymbol{X}},\boldsymbol{y},\boldsymbol{\theta}\big)\,\middle|\,\boldsymbol{y}\right]\\ &\approx\frac{\partial E_{\mathrm{recog}}}{\partial\hat{\boldsymbol{x}}}\Big(\mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{y}}\big[\check{\boldsymbol{X}}\,\big|\,\boldsymbol{y}\big],\boldsymbol{y},\boldsymbol{\theta}\Big)\\ &=\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\nu}_{\boldsymbol{y}}}\left(\frac{\lambda}{2}\Big\|\boldsymbol{y}-\mathbf{C}\boldsymbol{\nu}_{\boldsymbol{y}}\Big\|^2+\boldsymbol{\alpha}^{\mathrm{T}}\Big|\boldsymbol{\nu}_{\boldsymbol{y}}\Big|\right)\overset{\text{set}}{=}0\\ \implies\boldsymbol{\nu}_{\boldsymbol{y}}^{*}&=\underset{\boldsymbol{\nu}_{\boldsymbol{y}}}{\mathrm{argmin}}\left\{\frac{\lambda}{2}\Big\|\boldsymbol{y}-\mathbf{C}\boldsymbol{\nu}_{\boldsymbol{y}}\Big\|^2+\boldsymbol{\alpha}^{\mathrm{T}}\Big|\boldsymbol{\nu}_{\boldsymbol{y}}\Big|\right\}.\end{aligned} \tag{6.21}$$

Thus[2], the optimal choice for the mean is the solution to the L^1-penalized least-squares problem. For "fat" matrices $\mathbf{C}$, this is known as basis-pursuit denoising,[3] a convex optimization for which many solvers exist.

That leaves the covariance matrix, given by Eq. 6.15. Again we exploit the fact that the Hessian of the energy is a sub-cubic—indeed, constant—function of $\hat{\boldsymbol{x}}$, except at points where any $\hat{x}_k$ is zero. Therefore,

$$\begin{aligned}\mathbf{P}_{\boldsymbol{y}} &= \mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{y}}\left[\frac{\partial^2 E_{\text{recog}}}{\partial\hat{\boldsymbol{x}}\,\partial\hat{\boldsymbol{x}}^{\mathrm{T}}}(\check{\boldsymbol{X}},\boldsymbol{y},\boldsymbol{\theta})\,\middle|\,\boldsymbol{y}\right] \\ &\approx \frac{\partial^2 E_{\text{recog}}}{\partial\hat{\boldsymbol{x}}\,\partial\hat{\boldsymbol{x}}^{\mathrm{T}}}\Big(\mathbb{E}_{\check{\boldsymbol{X}}|\boldsymbol{y}}\big[\check{\boldsymbol{X}}\,\big|\,\boldsymbol{y}\big],\boldsymbol{y},\boldsymbol{\theta}\Big) \\ &\approx \lambda\mathbf{C}^{\mathrm{T}}\mathbf{C} + \operatorname{diag}\Big(\boldsymbol{\alpha}\circ\beta\operatorname{sech}^2(\beta\boldsymbol{\nu}_{\boldsymbol{y}})\Big).\end{aligned} \tag{6.22}$$

The approximation in the final line follows from the approximation to the Laplace energy in Eq. 6.17.

Let us collect up our optimizations for the complete sparse-coding algorithm:

EM Algorithm for Laplace-Sparse Coding

- **E step**: $\boldsymbol{\nu}_{\boldsymbol{y}}^{(i+1)} \overset{\text{set}}{=} \operatorname{argmin}_{\boldsymbol{\nu}_{\boldsymbol{y}}}\left\{\frac{\lambda}{2}\left\|\boldsymbol{y}-\mathbf{C}_i\boldsymbol{\nu}_{\boldsymbol{y}}\right\|^2 + \boldsymbol{\alpha}^{\mathrm{T}}\left|\boldsymbol{\nu}_{\boldsymbol{y}}\right|\right\}$

  $\mathbf{P}_{\boldsymbol{y}}^{(i+1)} \overset{\text{set}}{=} \lambda\mathbf{C}_i^{\mathrm{T}}\mathbf{C}_i + \operatorname{diag}\Big(\boldsymbol{\alpha}\circ\beta\operatorname{sech}^2(\beta\boldsymbol{\nu}_{\boldsymbol{y}}^{(i+1)})\Big)$
- **M step**: $\mathbf{C} \overset{\text{set}}{=} \left\langle \boldsymbol{Y}\boldsymbol{\nu}_{\boldsymbol{Y}}^{(i+1)^{\mathrm{T}}}\right\rangle_{\boldsymbol{Y}} \left\langle \mathbf{P}_{\boldsymbol{Y}}^{(i+1)^{-1}} + \boldsymbol{\nu}_{\boldsymbol{Y}}^{(i+1)}\boldsymbol{\nu}_{\boldsymbol{Y}}^{(i+1)^{\mathrm{T}}}\right\rangle_{\boldsymbol{Y}}^{-1}.$

Relationship to classical sparse coding. We have arrived at a sparse-coding algorithm very similar to that of Lewicki and colleagues [14, 16], but by a somewhat different route. In those papers, the matrix of basis functions, $\mathbf{C}$, is fit by directly descending along the gradient of the *marginal* cross entropy, $\mathrm{H}_{p\hat{p}}[\boldsymbol{Y};\mathbf{C}]$, rather than descending via the gradient of its upper bound, the free energy. The price, of course, is the intractability of the marginal cross entropy. In the classical papers, this price is paid with Laplace's method, approximating the model marginal, $\hat{p}\Big(\hat{\boldsymbol{Y}};\mathbf{C}\Big)$, with the integral $\int_{\hat{\boldsymbol{x}}}\hat{p}(\hat{\boldsymbol{x}})\hat{p}\Big(\hat{\boldsymbol{x}}\,\Big|\,\hat{\boldsymbol{Y}};\mathbf{C}\Big)\,\mathrm{d}\hat{\boldsymbol{x}}$ in the vicinity of one of the modes, $\hat{\boldsymbol{x}}^0$, of the model joint (Eq. 2.35). This in some sense removes the latent variables from the problem, although the modes still need to be estimated. Consequently, there is no E step, and no fixed parameters $\boldsymbol{\nu}_{\boldsymbol{y}}$ and $\mathbf{P}_{\boldsymbol{y}}$ (for any $\boldsymbol{y}$).

Nevertheless, the Hessian of the recognition (or joint) energy still shows up in the classical derivation's loss, this time as a byproduct of Laplace's method, as a measure of the volume of the recognition distribution. This is no coincidence: we have allowed the E step to collapse into a Laplace approximation [19] by using an energy function that is sub-cubic almost everywhere. However, in the classical method, this precision matrix is not fixed during optimization of $\mathbf{C}$, but rather is interpreted as a function of $\mathbf{C}$ and $\hat{\boldsymbol{x}}^0$:

[2]Technically the final line is only a sufficient, not a necessary, condition for the gradient to be zero, but we are assuming the Hessian is locally positive definite.

[3]This is also sometimes referred to as the LASSO problem [], although typically that term is reserved for "tall" output matrices $\mathbf{C}$.

$f(\mathbf{C}, \hat{\boldsymbol{x}}^0) := \log\left|\mathbf{P}_{\boldsymbol{y}}\left(\mathbf{C}, \hat{\boldsymbol{x}}^0\right)\right|$. On the other hand, the trace term in Eq. 6.19, $\lambda \mathrm{tr}\left[\mathbf{C}\mathbf{P}_{\boldsymbol{y}}^{-1}\mathbf{C}^{\mathrm{T}}\right]$, never materializes, because the squared reconstruction error is simply evaluated at the mode, rather than being averaged under a proxy recognition distribution.

Conceptually, these two terms encode the contribution of the recognition precision ($\mathbf{P}_{\boldsymbol{y}}$) to the two losses, ours and the classical. Evidently, increasing the precision has opposite effects on the two losses, lowering the free energy (via $\lambda \mathrm{tr}\left[\mathbf{C}\mathbf{P}_{\boldsymbol{y}}^{-1}\mathbf{C}^{\mathrm{T}}\right]$) but raising the approximate marginal cross entropy (via $\log\left|\mathbf{P}_{\boldsymbol{y}}\left(\mathbf{C}, \hat{\boldsymbol{x}}^0\right)\right|$). But increasing the magnitude of *the emission matrix* $\mathbf{C}$ has the *same* effect on these two terms: It increases the contribution of recognition uncertainty to the average reconstruction error ($\lambda \mathrm{tr}\left[\mathbf{C}\mathbf{P}_{\boldsymbol{y}}^{-1}\mathbf{C}^{\mathrm{T}}\right]$), and it increases the recognition precision itself (and therefore $\log\left|\mathbf{P}_{\boldsymbol{y}}\left(\mathbf{C}, \hat{\boldsymbol{x}}^0\right)\right|$) via Eq. 6.22. Indeed, the partial derivative of $f(\mathbf{C}, \hat{\boldsymbol{x}}^0) := \log\left|\mathbf{P}_{\boldsymbol{y}}\left(\mathbf{C}, \hat{\boldsymbol{x}}^0\right)\right|$ with respect to $\mathbf{C}$ is precisely the same as the derivative of the trace term $\lambda \mathrm{tr}\left[\mathbf{C}\mathbf{P}_{\boldsymbol{y}}^{-1}\mathbf{C}^{\mathrm{T}}\right]$ with respect to $\mathbf{C}$ (namely, $\lambda \mathbf{C}\mathbf{P}_{\boldsymbol{y}}^{-1}$).

There is one last discrepancy between the loss gradients. In the classical derivation, the mode $\hat{\boldsymbol{x}}^0$ is not fixed, either, and in turns depends on $\mathbf{C}$. So the *total* derivative of $\log\left|\mathbf{P}_{\boldsymbol{y}}\left(\mathbf{C}, \hat{\boldsymbol{x}}^0(\mathbf{C})\right)\right|$ generates extra terms. On the other hand, in the original papers, these are derived and then promptly dropped, under the conjecture that curvature unrelated to volume is irrelevant to the optimization.

Independent-components analysis

Just as EM becomes K-means for the Gaussian mixture model and (iterative) principal-components analysis for the factor analyzer, as the variance of the (Gaussian) emission is shrunk to zero, EM for the sparse coding model becomes independent-components analysis (ICA).....

6.5.2 Variational Autoencoders

The use of a separate posterior distribution for every observation, $\boldsymbol{y}$, will clearly fail if the number of data is too large. An alternative is to encourage the recognition distribution to use a single set of parameters for all data, by specifying a single, parameter-dependent mapping from the observations to the moments of the proxy recognition distribution. For example, sticking with Gaussian recognition distributions, we can use

$$\check{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}\big) = \mathcal{N}\big(\boldsymbol{\nu}(\boldsymbol{Y}, \boldsymbol{\phi}), \mathbf{P}^{-1}(\boldsymbol{Y}, \boldsymbol{\phi})\big).$$

The moments, $\boldsymbol{\nu}(\boldsymbol{Y}, \boldsymbol{\phi})$ and $\mathbf{P}^{-1}(\boldsymbol{Y}, \boldsymbol{\phi})$, are now interpreted to be mappings or functions rather than parameters. For example, the functions can be deep neural networks, and so in some sense arbitrarily flexible. For sufficiently large $\boldsymbol{\phi}$, this flexibility could include (depending on the ratio of parameters to observations) the ability to store a separate mean and precision for every datum, and therefore to mimic Eq. 6.13, but ideally it does not. Limiting the flexibility of the functions allows for the computational cost of learning $\boldsymbol{\phi}$ to be "amortized" across observations, and for the model to provide good recognition distributions for new observations $\boldsymbol{y}$.

6.5.3 Diffusion Models

6.6 Proxy recognition distributions without parameters: Variational inference

6.7 Non-random "latent" variables.

To reiterate the central theme of this chapter: the fundamental trade-off for generative models is between expressive power and ease of inference. So far we have explored three different strategies for trading off: Severely limit expressive power to conjugate or pseudo-conjugate source distributions in order to allow for exact inference (Section 6.4); allow arbitrarily expressive generative models, but then approximate inference with a recognition distribution that is either arbitrary, but highly parameterized (Section 6.5); or correct up to simplifying but erroneous independence assumptions (Section 6.6). Now we introduce a fourth strategy: let the latent variables of the model be related to the observed variables by an invertible (and therefore deterministic) transformation. This makes the model marginal $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$ computable in closed-form with the standard rules of calculus for changing variables under an integral. Consequently, the marginal cross entropy, $\mathrm{H}_{p\hat{p}}[\boldsymbol{Y};\boldsymbol{\theta}]$, can be descended directly, rather than indirectly via the free energy, and EM is not needed.

Below we explore two such models in the usual way, starting with simple linear transformations and then moving to more complicated functions. But let us begin with a more abstract formulation, in order to make contact with the unsupervised, discriminative learning problems of Section 5.2. In that section, we related observations $\hat{\boldsymbol{Y}}$ to (hypothetical) latent variables $\hat{\boldsymbol{Z}}$ via a deterministic, invertible transformation, Eq. 5.25. Here we shall make a similar specification, although to emphasize that this is a generative model, we write the observations as a function of the latent variables, $\hat{\boldsymbol{X}}$, rather than vice versa. Still, to exhibit the relationship with the discriminative model, we denote this function as the inverse of some recognition function, $\boldsymbol{g}(\cdot,\boldsymbol{\theta})$:

$$\hat{\boldsymbol{Y}} = \boldsymbol{g}^{-1}(\hat{\boldsymbol{X}},\boldsymbol{\theta}). \tag{6.23}$$

For generative models, it is also necessary to specify a distribution over the latent source variables. Although the discriminative model makes no such specification, its training objective—maximization of the latent ("output") entropy—will tend to produce independent outputs. Therefore, we shall *assume* that the generative model's latent variables are independent—and, for now, nothing else:

$$\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big) = \prod_{k=1}^{K} \hat{p}_{\hat{X}_k}\Big(\hat{X}_k\Big) = \prod_{k=1}^{K} \frac{\mathrm{d}\psi_k}{\mathrm{d}\hat{x}_k}\Big(\hat{X}_k\Big). \tag{6.24}$$

In the second equality, we have simply expressed the probability distributions in terms of the cumulative distribution functions, $\psi_k(\hat{X}_k)$, or more precisely their derivatives, in anticipation of the results below.

In the discriminative models of Section 5.2, rather than being specified explicitly, the distribution of latent variables was inherited from the data distribution, $p(\boldsymbol{Y})$, via the change-of-variables formula. Here something like the reverse obtains: The distribution of the (model) observations $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$ is inherited, via the deterministic transformation Eq.

6.23 and the change-of-variables formula, from the distribution of latent variables, Eq. 6.24:

$$\begin{aligned}
\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) &= \hat{p}\Big(\boldsymbol{g}(\hat{\boldsymbol{Y}},\boldsymbol{\theta});\boldsymbol{\theta}\Big)\left|\frac{\mathrm{d}\boldsymbol{g}}{\mathrm{d}\hat{\boldsymbol{y}}}(\hat{\boldsymbol{Y}},\boldsymbol{\theta})\right| \\
&= \prod_{k=1}^{K}\frac{\mathrm{d}\psi_k}{\mathrm{d}\hat{x}_k}\Big(g_k(\hat{\boldsymbol{Y}},\boldsymbol{\theta})\Big)\left|\frac{\mathrm{d}\boldsymbol{g}}{\mathrm{d}\hat{\boldsymbol{y}}}(\hat{\boldsymbol{Y}},\boldsymbol{\theta})\right| \\
&= \left|\frac{\mathrm{d}\boldsymbol{\psi}}{\mathrm{d}\hat{\boldsymbol{x}}}\Big(\boldsymbol{g}(\hat{\boldsymbol{Y}},\boldsymbol{\theta})\Big)\right|\left|\frac{\mathrm{d}\boldsymbol{g}}{\mathrm{d}\hat{\boldsymbol{y}}}(\hat{\boldsymbol{Y}},\boldsymbol{\theta})\right| \\
&= \left|\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\hat{\boldsymbol{y}}}(\hat{\boldsymbol{Y}},\boldsymbol{\theta})\right|.
\end{aligned} \tag{6.25}$$

The third line follows from the fact that $\mathrm{d}\boldsymbol{\psi}/\,\mathrm{d}\hat{\boldsymbol{x}}$ is a diagonal matrix, so its determinant is just the product of the diagonal elements. In the final line, we have combined the two determinants using

$$\boldsymbol{f}\Big(\hat{\boldsymbol{Y}}\Big) := \boldsymbol{\psi}\Big(\boldsymbol{g}\Big(\hat{\boldsymbol{Y}}\Big)\Big).$$

We have recovered Eq. 5.28, but from a different model [2]. In the discriminative model, the Jacobian $\mathrm{d}\boldsymbol{f}/\,\mathrm{d}\hat{\boldsymbol{y}}$ results from an explicitly defined input-output relationship, $\hat{\boldsymbol{Z}} = \boldsymbol{\psi}(\boldsymbol{Y},\boldsymbol{\theta})$. In the generative model defined here, $\mathrm{d}\boldsymbol{\psi}/\,\mathrm{d}\hat{\boldsymbol{y}}$ is the result of the composition of *two* functions, an input-output relationship $\hat{\boldsymbol{X}} = \boldsymbol{g}(\hat{\boldsymbol{Y}},\boldsymbol{\theta})$, and the cumulative distribution functions of the latent variables, $\boldsymbol{\psi}(\hat{\boldsymbol{X}})$. Thus the "outputs" $\hat{\boldsymbol{Z}}$ of the discriminative model are the latent variables of the generative model *passed through their own cdfs*. If we define $\hat{\boldsymbol{Z}} := \boldsymbol{\psi}(\hat{\boldsymbol{X}})$ for the generative model, then we can say that the generative $\hat{\boldsymbol{Z}}$ are distributed independently (because $\hat{\boldsymbol{X}}$ are independent and $\boldsymbol{\psi}(\cdot)$ acts elementwise) and uniformly (because the cdfs exactly flatten the distribution of $\hat{\boldsymbol{X}}$). This is consistent with the discriminative objective: maximizing the entropy of $\hat{\boldsymbol{Z}}$ will likewise tend to distribute them independently and uniformly.

Thus, *in the deterministic, invertible setting, density estimation in a generative model is equivalent to maximizing mutual information through a discriminative model.* It might seem that this equivalence holds only for a subset of possible discriminative models: Eq. 6.25 holds for Jacobian determinants $\left|\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\hat{\boldsymbol{y}}}(\hat{\boldsymbol{Y}},\boldsymbol{\theta})\right|$ that can be factored as in the penultimate line, $\left|\frac{\mathrm{d}\boldsymbol{\psi}}{\mathrm{d}\hat{\boldsymbol{x}}}\Big(\boldsymbol{g}(\hat{\boldsymbol{Y}},\boldsymbol{\theta})\Big)\right|\left|\frac{\mathrm{d}\boldsymbol{g}}{\mathrm{d}\hat{\boldsymbol{y}}}(\hat{\boldsymbol{Y}},\boldsymbol{\theta})\right|$, with $\frac{\mathrm{d}\boldsymbol{\psi}}{\mathrm{d}\hat{\boldsymbol{x}}}\Big(\boldsymbol{g}(\hat{\boldsymbol{Y}},\boldsymbol{\theta})\Big)$ diagonal; whereas Eq. 5.28 holds for all invertible functions $\boldsymbol{f}(\hat{\boldsymbol{X}})$. But this is a consequence of a non-essential, albeit useful, restriction imposed on the generative model in Eqs. 6.23 and 6.24: that the sources be independent. As long as the latent variables $\hat{\boldsymbol{X}}$ are uniformly distributed, $\hat{p}\Big(\boldsymbol{g}(\hat{\boldsymbol{Y}},\boldsymbol{\theta});\boldsymbol{\theta}\Big)$ reduces to a constant, and the final line of Eq. 6.25 follows immediately from the first (with $\boldsymbol{g}$ in place of $\boldsymbol{f}$).

6.7.1 InfoMax ICA, revisited

This equivalence was noted first historically [2] for a specific model, InfoMax ICA [1], which we first encountered in Section 5.2. In this simplest case, the observations are related to the "latent" variables by a square, full-rank matrix:

$$\hat{\boldsymbol{Y}} = \mathbf{C}\hat{\boldsymbol{X}} = \mathbf{G}^{-1}\hat{\boldsymbol{X}}.$$

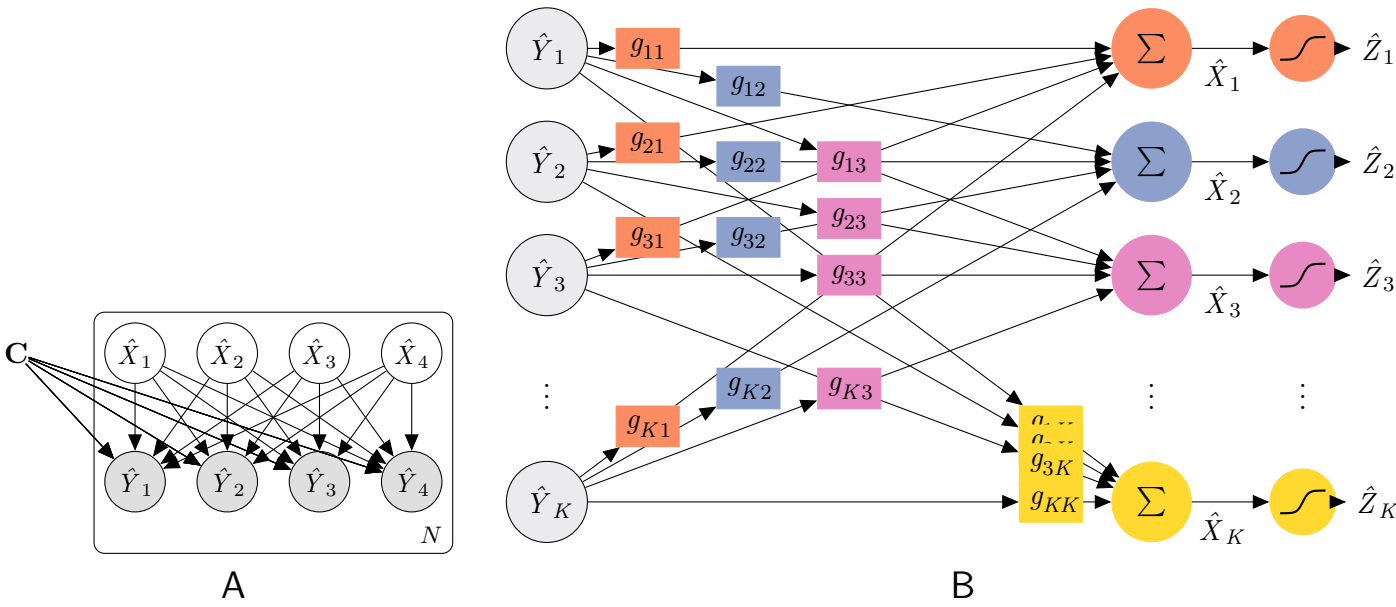


Figure 6.1: **"Infomax" independent-components analysis.** (A) The generative model. (B) The discriminative model. Note that $\mathbf{C} = \mathbf{G}^{-1}$.

Substituting this relationship (cf. Eq. 6.23) into Eq. 6.25, we see that the marginal distribution of the observed variables is

$$\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big) = \left|\frac{\mathrm{d}\boldsymbol{\psi}}{\mathrm{d}\hat{\boldsymbol{x}}}\Big(\mathbf{G}\,\hat{\boldsymbol{Y}}\Big)\right|\mathbf{G},$$

where again $\boldsymbol{\psi}$ is the array of cdfs for each of the independent source variables. As a consequence of the discriminative-generative duality, fitting this marginal density follows the same gradient as in InfoMax ICA, Eq. 5.31,

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{G}}\,\mathrm{D}_{\mathrm{KL}}\left\{p(\boldsymbol{Y})\middle|\middle|\left|\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{y}}(\boldsymbol{Y},\mathbf{G})\right|\right\} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{G}}\mathbb{E}_{\boldsymbol{Y}}\left[-\log\left|\frac{\mathrm{d}\boldsymbol{\psi}}{\mathrm{d}\hat{\boldsymbol{x}}}(\mathbf{G}\,\boldsymbol{Y})\right|\right] - \mathbf{G}^{-\mathrm{T}}.$$

That is, InfoMax ICA can be implemented as density estimation in a generative model with sources distributed independently and cumulatively according to $\boldsymbol{\psi}$ [2]; see Fig. 6.1.

But we haven't specified $\boldsymbol{\psi}$! This omission may have seemed minor in the discriminative model—sigmoidal nonlinearities in neural networks are typically selected rather freely—but is striking in a generative model. And indeed, the choice matters. Suppose we had let the sigmoidal function be the cdf of a Gaussian. Then independent outputs $\hat{\boldsymbol{Z}}$ could be achieved merely by decorrelating the inputs, since for jointly Gaussian random variables, uncorrelatedness implies independence. In contrast, the (standard) logistic function is super-Gaussian (leptokurtotic), so InfoMax ICA with logistic outputs will generally do more than decorrelate its inputs. This may seem remarkable, given the visually minor discrepancy between the Gaussian cdf and the logistic function (Fig. **??**; B.A. Olshausen, personal communication). Now we see the advantage of the generative perspective, from which this difference is more salient—and at long last, (partially) answer the question of how to choose the feedforward nonlinearity, $\boldsymbol{\psi}$, in InfoMax ICA.

6.7.2 Nonlinear independent-component estimation

[[XXX]]

An alternative view of the E step.

Greater insight into the algorithm can be had by closer investigation into its apparent symmetry. Even though both steps can be written as minimizations of the same quantity, the free energy, whereas the M step is generally implemented via a minimization of the complete cross entropy (as discussed in the previous section), the E step cannot be: Minimizing the free energy with respect to $\boldsymbol{\phi}$ is *not* in general the same as minimizing the complete cross entropy with respect to $\boldsymbol{\phi}$—even though, as we saw, this is the case for $\boldsymbol{\theta}$. To wit:

$$\begin{aligned}\underset{\boldsymbol{\phi}}{\operatorname{argmin}}\{F(\boldsymbol{\theta}_t, \boldsymbol{\phi})\} &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\Big\{\big\langle \log \check{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}\big) - \log \hat{p}\big(\check{\boldsymbol{X}},\, \boldsymbol{Y}; \boldsymbol{\theta}_t\big)\big\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}\Big\} \\ &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\Big\{\big\langle \log \check{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}\big)\big\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}} + \big\langle -\log \hat{p}\big(\check{\boldsymbol{X}},\, \boldsymbol{Y}; \boldsymbol{\theta}_t\big)\big\rangle_{\check{\boldsymbol{X}}, \boldsymbol{Y}}\Big\} \\ &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\big\{-\mathrm{H}_{\check{p}}\big[\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}; \boldsymbol{\phi}\big] + \mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}},\, \boldsymbol{Y}; \boldsymbol{\theta}_t, \boldsymbol{\phi}\big]\big\},\end{aligned} \tag{6.26}$$

where $\mathrm{H}_{\check{p}}\big[\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}; \boldsymbol{\phi}\big]$ is the conditional entropy of the proxy recognition distribution. This yields an alternative interpretation of the E step: parameters are adjusted so as to reduce the complete cross entropy, as in the M step, but balanced against the cost of reducing the entropy of the proxy recognition distribution. However, improvements in the complete cross entropy are achieved not by fitting the observed data better (as in the M step), but by averaging the joint under a superior recognition distribution—essentially, by fitting the latent variables better. So the ideal recognition distribution specifies the latent variables as minimally as possible (minimizing the negative entropy), but distributes those variables in a way most congruent with the observed data (minimizing the complete cross entropy). Intuitively, the first term prevents the second term from enforcing a deterministic recognition distribution that takes no account of the uncertainty in the actual joint distribution of $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$. We shall make use of this interpretation in our discussion in the following section of the two main approaches to solving Eq. 6.26.

Finally, note that although minimizing free energy in the E step is not equivalent to minimizing complete cross entropy, it is still (as in the M step) equivalent to minimizing the conditional KL divergence between proxy and true recognition distributions, $\mathrm{D}_{\mathrm{KL}}\big\{\check{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}\big) \big|\big| \hat{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}; \boldsymbol{\theta}_t\big)\big\}$, as in our first interpretation of the E step. However, it is not in fact possible to implement the E step explicitly in these terms: Recall that the point of using a proxy recognition distribution, $\check{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}\big)$, was that the true recognition distribution $\hat{p}\Big(\hat{\boldsymbol{X}} \,\Big|\, \hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big)$ was supposed to be impossible to compute—yet the latter appears explicitly in the above expression, which consequently cannot be used.

In fine, the M step can be interpreted as minimizing, with respect to $\boldsymbol{\theta}$, the complete cross entropy, the free energy, and the KL divergence between two joint distributions. In practice, it is implemented as a minimization of the complete cross entropy. The E step can be interpreted as minimizing, with respect to $\boldsymbol{\phi}$, only the free energy and a (conditional) KL divergence, in this case between the true and proxy recognition distributions. How it is implemented in practice we consider next.

First- and second-order varieties of general EM.

The various implementations of this more general version of EM essentially turn on what kind of function we use for the proxy recognition distribution; that determines what the minimization in Eq. 6.26 looks like. In the simplest, "*variational* EM," the recognition distribution is approximated by a Dirac delta at some function of the observed variables and parameters, $\boldsymbol{\phi}(\boldsymbol{y})$:

$$\check{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y}\big) = \delta(\check{\boldsymbol{X}} - \boldsymbol{\phi}(\boldsymbol{Y})).$$

Recall that the delta function has zero entropy. This amounts, then, under the interpretation of the E step lately made, to giving up on the first term in Eq. 6.26, the conditional entropy of the proxy recognition distribution, in order to maximize the second term, the complete cross entropy. Thus Eq. 6.26 becomes:

$$\begin{aligned}
\boldsymbol{\phi}_{t+1} &= \operatorname*{argmin}_{\boldsymbol{\phi}} \big\{ -\mathrm{H}_{\check{p}}\big[\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y};\boldsymbol{\phi}\big] + \mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}},\,\boldsymbol{Y};\boldsymbol{\theta}_t,\boldsymbol{\phi}\big] \big\} \\
&= \operatorname*{argmin}_{\boldsymbol{\phi}} \big\{ \big\langle -\log \hat{p}(\boldsymbol{\phi}(\boldsymbol{Y}),\,\boldsymbol{Y};\boldsymbol{\theta}_t) \big\rangle_{\boldsymbol{Y}} \big\} \\
&= \operatorname*{argmin}_{\boldsymbol{\phi}} \big\{ \big\langle -\log \hat{p}(\boldsymbol{\phi}(\boldsymbol{Y})|\,\boldsymbol{Y};\boldsymbol{\theta}_t) - \log \hat{p}(\,\boldsymbol{Y};\boldsymbol{\theta}_t) \big\rangle_{\boldsymbol{Y}} \big\} \\
&= \operatorname*{argmin}_{\boldsymbol{\phi}} \big\{ \big\langle -\log \hat{p}(\boldsymbol{\phi}(\boldsymbol{Y})|\,\boldsymbol{Y};\boldsymbol{\theta}_t) \big\rangle_{\boldsymbol{Y}} \big\}
\end{aligned} \tag{6.27}$$

It is more or less clear that for each observation $\boldsymbol{y}$, we should choose $\boldsymbol{\phi}(\boldsymbol{y})$ to be the mode of the recognition distribution, $\boldsymbol{\phi}(\boldsymbol{y}) \overset{\text{set}}{=} \operatorname{argmax}_{\hat{\boldsymbol{x}}}\{\hat{p}(\hat{\boldsymbol{x}}|\boldsymbol{y};\boldsymbol{\theta}_t)\}$, although this can be shown more formally using the calculus of variations (see below). This variational approach, then, amounts simply to approximating the recognition distribution with a delta function at its peak. Essentially, this choice of proxy recognition distribution makes no allowance for the value of $\check{\boldsymbol{x}}$ it recommends being wrong, so it pays a high price in terms of the negative conditional entropy; but it partially compensates for this by choosing that value very well, and hence minimizing the complete cross entropy.

NB: in variational EM, we optimize a *functional* rather than a function, because we are looking for the optimal function $\phi(Y)$ rather than an optimal value. That is why this is a problem for the calculus of variations. Here you might introduce the ideas that will become useful for EM under sparse coding... Then discuss mean-field VI (CAVI; coordinate ascent variational inference). See Blei's review paper. Then include your discussion of recent developments (variational autoencoders, NICE, normalizing flows, etc.).

More generally, we may select some parametric form for $\check{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y}\big)$—an isotropic Gaussian is very common—and optimize Eq. 6.26 directly by differentiating with respect to those parameters. (The isotropy reduces the number of parameters in the normal distribution to $2N$, where N is the number of latent variables, as well as making the recognition distribution factorizable.) Alternatively, if we can at least sample $\hat{\boldsymbol{x}}$ from (if not compute) $\hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\boldsymbol{y};\boldsymbol{\theta}\Big)$, for various $\boldsymbol{y}$ from $p(\,\boldsymbol{Y})$, then we can calculate sample cumulants and set our proxy recognition distribution's cumulants equal to these. This requires assuming a functional form for the relationship between $\boldsymbol{y}$ and the cumulants of the recognition distribution. If the isotropy is to be maintained, it also requires (in every E step) computing the eigendecomposition of the sampled data and then replacing $\hat{\boldsymbol{X}}$ with the rotated/scaled version that is most isotropic. For example, if the recognition covariance is assumed to be independent of $\hat{\boldsymbol{Y}}$, one could estimate $\mathrm{Cov}_{\hat{\boldsymbol{X}}|\,\hat{\boldsymbol{Y}}}\Big[\hat{\boldsymbol{X}}|\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big|\,\hat{\boldsymbol{Y}}\Big] =: \Sigma$ from the samples, and then replace $\hat{\boldsymbol{X}}$ with $\Sigma^{-1/2}\hat{\boldsymbol{X}}$ in the proxy recognition distribution. (The model distribution would also have to be updated accordingly.) [[citation??]]

The true model recognition distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$, will not in general be factorizable, let alone Gaussian, yet such proxy recognition distributions work well in practice. This choice of proxy recognition distribution also lends itself well to the interpretation of the E step just suggested: The complete cross entropy is kept small by choosing a "good"

mean/mode—i.e., one that the model would have inferred from the data (which is just the inference we can't compute); and "good" variances—i.e., ones that are close to the uncertainty of the model's own inferences. But the variances are also encouraged simply to grow larger by the penalty on the recognition distribution's negative entropy.

6.7.3 The relationship between the complete and incomplete cross entropies

This section should be cleaned up, maybe rewritten, after the preceding section has been.

The entropy of the recognition distribution. EM relaxes the problem of minimizing the incomplete cross-entropy—maximizing the fit of the model to the observed data—in *two* different ways:

1. Use of the merely "approximate" recognition distribution, $\check{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}\big)$—even when this is simply the true recognition distribution evaluated at the parameters of the previous time step ("standard EM")—in place of the true model recognition distribution, $\hat{p}\Big(\hat{\boldsymbol{X}} \,\Big|\, \hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big)$. We saw that this approximation results in an iterative, rather than one-shot, algorithm.

2. Use of the complete rather than (what we really care about) the incomplete cross entropy.

It's perhaps harder to appreciate the implications of 2, so we consider it in isolation from 1. That is, the proxy recognition distribution was adopted because, not depending directly on $\boldsymbol{\theta}$, it renders the calculation of the gradient in Eq. 6.3 tractable; but we shall now ignore this intractability and drop the use of a proxy recognition distribution, in order to examine alone the effect of 2.

From Eq. 6.3 it is clear that the gradients of the complete and incomplete cross entropies are not equal (the derivative with respect to $\boldsymbol{\theta}$ in the final line can't be pulled outside the averaging brackets, which also depend on $\boldsymbol{\theta}$). What is their relationship? To answer this question, we use the *true* recognition distribution in the averaging distribution:

$$\check{p}\big(\check{\boldsymbol{X}},\, \boldsymbol{Y}\big) \overset{\text{set}}{=} \check{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}\big) p(\,\boldsymbol{Y}),$$

in which case the complete cross entropy becomes:

$$\mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}},\, \boldsymbol{Y}; \boldsymbol{\theta}, \boldsymbol{\theta}\big] = \mathrm{H}_{p\hat{p}}[\,\boldsymbol{Y}; \boldsymbol{\theta}] + \mathrm{H}_{\hat{p}}\big[\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}; \boldsymbol{\theta}\big]. \tag{6.28}$$

In words, the difference between the complete and incomplete cross entropies is simply the entropy of the recognition distribution, averaged under the data distribution.[4] Notice that this term (the last) is precisely the first term in the last line of Eq. 6.26, evaluated at $\check{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}\big) = \hat{p}\big(\check{\boldsymbol{X}} \,\big|\, \boldsymbol{Y}; \boldsymbol{\theta}\big)$. Colloquially, we can say that the less specific the model is in the inferences to $\check{\boldsymbol{x}}$ that it licenses, the looser the bound that the complete cross entropy provides on the incomplete cross entropy.

To get a better grip on this idea, let us consider the two extremes. Let $\mathrm{H}_{\hat{p}}\Big[\hat{\boldsymbol{X}}\Big]$ be the entropy of the model latent variables. Conditioning on $\boldsymbol{Y}$ cannot increase this entropy, so $0 \leq \mathrm{H}_{\hat{p}}\Big[\hat{\boldsymbol{X}} \,\Big|\, \boldsymbol{Y}; \boldsymbol{\theta}\Big] \leq \mathrm{H}_{\hat{p}}\Big[\hat{\boldsymbol{X}}\Big]$. At the first extreme, $\mathrm{H}_{\hat{p}}\Big[\hat{\boldsymbol{X}} \,\Big|\, \boldsymbol{Y}; \boldsymbol{\theta}\Big] = 0$, and the relationship

[4] This conditional entropy should not be confused with $\mathrm{H}_{\hat{p}}\Big[\hat{\boldsymbol{X}} \,\big|\, \hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big]$, the average of the recognition distribution under the *model* marginal, $\hat{p}\Big(\hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big)$.

between $\boldsymbol{Y}$ and $\hat{\boldsymbol{X}}$ becomes deterministic. Intuitively, it is always dead obvious which $\hat{\boldsymbol{x}}$ is responsible for each vector of observations $\boldsymbol{y}$. In this case, reducing the complete cross entropy is precisely equivalent to reducing the incomplete cross entropy, and in fact EM may become unnecessary. (We shall shortly see an example of such models in our discussion of ICA and its nonlinear counterparts.) This might make it seem desirable for the model *always* to learn a deterministic recognition distribution. But the price: ...

At the other extreme, $\mathrm{H}_{\hat{p}}\Big[\hat{\boldsymbol{X}}\Big|\boldsymbol{Y};\boldsymbol{\theta}\Big] = \mathrm{H}_{\hat{p}}\Big[\hat{\boldsymbol{X}}\Big]$. This can happen, for example, if we made a mistake in using a latent-variable approach—perhaps the data are simply normally distributed. Then a good model will not discover any mutual information between $\boldsymbol{Y}$ and $\hat{\boldsymbol{X}}$, and we can set $\mathrm{H}_{\hat{p}}\Big[\hat{\boldsymbol{X}}\Big|\boldsymbol{Y};\boldsymbol{\theta}\Big]$ to zero without any offsetting penalty. Then reducing the complete cross entropy is again equivalent to reducing the incomplete cross entropy. In contrast to the previous case.....

Outside of such pathological cases, the last term will never be zero, so the complete and incomplete cross entropies will never be equal.

6.7.4 Minimum description length and the "bits-back" argument

The sender-receiver game. There is another way to think about the free energy [11], most intuitively expressed in terms of the Gaussian mixture model (see Section 2.1.1 and again Section 6.4.1 below, and Fig. **??**). Imagine that I (the "sender") want to communicate an observed datum, $\boldsymbol{y}$, to someone else (the "receiver"). Ideally, I would encode this datum according to the model marginal distribution $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$ (or even more ideally, the data marginal $p(\boldsymbol{Y})$), and pay (on average) $\mathrm{H}_{p\hat{p}}[\boldsymbol{Y};\boldsymbol{\theta}]$ bits. But by the hypothesis we have been working under throughout, we cannot get a analytic expression for this quantity.

Put a figure of GMM data here!

Alternatively, assuming I have learned a good generative model for the data ($\hat{p}(\boldsymbol{Y};\boldsymbol{\theta}) = p(\boldsymbol{Y})$), I ought to be able to use it to transmit $\boldsymbol{y}$ efficiently. Intuitively, I should be able to assign the observation $\boldsymbol{y}$ to a class, $\hat{\boldsymbol{x}}$, and then transmit this class $\hat{\boldsymbol{x}}$ along with the data "misfit," i.e. the difference between $\boldsymbol{y}$ and the class mean. The receiver can use the class assignment, $\hat{\boldsymbol{x}}$, along with the data misfits to reconstruct $\boldsymbol{y}$. (Since the receiver must know my encoding scheme in order to recover these, I also pay a one-time cost to communicate the generative model to her. We will assume this is small compared with the data to be communicated and ignore it in what follows.)

To work out the costs of such a scheme, we need to consider precisely how we assign the observation ($\boldsymbol{y}$) to a class ($\hat{\boldsymbol{x}}$). *Prima facie*, the optimal candidate class might seem to be the peak of the recognition distribution, conditioned on the observation: $\mathrm{argmax}_{\hat{\boldsymbol{x}}}\{\hat{p}(\hat{\boldsymbol{x}}|\boldsymbol{y};\boldsymbol{\theta})\}$—in a GMM, the cluster to which the observed datum is most likely to belong. Assuming we design our encoding optimally, i.e. according to the model source distribution, $\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)$, the average optimal cost of encoding classes assigned in this way is the average of the surprisal, $-\log\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)$, under a Dirac delta situated at the mode of the recognition distribution, conditioned on the observation, all averaged under the (data) distribution of observations. If we call this averaging distribution $\check{p}\big(\check{\boldsymbol{X}},\,\boldsymbol{Y}\big) := \delta\Big(\check{\boldsymbol{X}} - \mathrm{argmax}_{\hat{\boldsymbol{x}}}\{\hat{p}(\hat{\boldsymbol{x}}\,|\,\boldsymbol{Y};\boldsymbol{\theta})\}\Big)p(\boldsymbol{Y})$, then we can call the cost $\mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}};\boldsymbol{\theta}\big]$. The average optimal cost of encoding the data misfit, in turn, is the surprisal of the emission distribution, $-\log\hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)$, averaged under the same joint distribution; call it $\mathrm{H}_{\check{p}\hat{p}}\big[\boldsymbol{Y}|\check{\boldsymbol{X}};\boldsymbol{\theta}\big]$. Thus the total cost is $\mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}};\boldsymbol{\theta}\big] + \mathrm{H}_{\check{p}\hat{p}}\big[\boldsymbol{Y}|\check{\boldsymbol{X}};\boldsymbol{\theta}\big]$.

The excess cost of hard cluster assignments. How does this compare to the optimal cost of using the model marginal? To find out, we rewrite the latter in terms of the model recognition, source, and emission distributions:

$$\begin{aligned}
\mathrm{H}_{p\hat{p}}[\boldsymbol{Y};\boldsymbol{\theta}] &= \left\langle -\log \hat{p}(\boldsymbol{Y};\boldsymbol{\theta})\right\rangle_{\boldsymbol{Y}} \\
&= \left\langle \left\langle -\log \hat{p}(\boldsymbol{Y};\boldsymbol{\theta})\right\rangle_{\check{\boldsymbol{X}}\,|\,\boldsymbol{Y}} \right\rangle_{\boldsymbol{Y}} \\
&= \left\langle \left\langle -\log \frac{\hat{p}(\check{\boldsymbol{X}};\boldsymbol{\theta})\hat{p}(\boldsymbol{Y}\,|\,\check{\boldsymbol{X}};\boldsymbol{\theta})}{\hat{p}(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y};\boldsymbol{\theta})} \right\rangle_{\check{\boldsymbol{X}}\,|\,\boldsymbol{Y}} \right\rangle_{\boldsymbol{Y}} \\
&= \mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}};\boldsymbol{\theta},\boldsymbol{\phi}\big] + \mathrm{H}_{\check{p}\hat{p}}\big[\boldsymbol{Y}\,|\,\check{\boldsymbol{X}};\boldsymbol{\theta},\boldsymbol{\phi}\big] - \mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}}\,|\,\boldsymbol{Y};\boldsymbol{\theta},\boldsymbol{\phi}\big].
\end{aligned} \tag{6.29}$$

Here we suppose $\check{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y}\big)$ to be some generic recognition distribution, e.g. optimized during the E step of general EM. It could, for example, be the actual model recognition distribution, $\check{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y}\big) = \hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$; we shall consider this shortly. But suppose (as we have been) that we have no access to $\hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$. Accordingly, we try to replace the model-recognition cross entropy, $\mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}}\,|\,\boldsymbol{Y};\boldsymbol{\theta},\boldsymbol{\phi}\big]$, in Eq. 6.29 with the proxy-recognition entropy, $\mathrm{H}_{\check{p}}\big[\check{\boldsymbol{X}}\,|\,\boldsymbol{Y};\boldsymbol{\phi}\big]$. Adding their relative entropy to both sides, we have

$$\begin{aligned}
&\mathrm{H}_{p\hat{p}}[\boldsymbol{Y};\boldsymbol{\theta}] + \mathrm{D}_{\mathrm{KL}}\big\{\check{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y}\big)\big|\big|\hat{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y};\boldsymbol{\theta}\big)\big\} \\
&\qquad = \mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}};\boldsymbol{\theta},\boldsymbol{\phi}\big] + \mathrm{H}_{\check{p}\hat{p}}\big[\boldsymbol{Y}\,|\,\check{\boldsymbol{X}};\boldsymbol{\theta},\boldsymbol{\phi}\big] - \mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}}\,|\,\boldsymbol{Y};\boldsymbol{\theta},\boldsymbol{\phi}\big] + \mathrm{D}_{\mathrm{KL}}\big\{\check{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y}\big)\big|\big|\hat{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y};\boldsymbol{\theta}\big)\big\} \\
&\qquad = \mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}};\boldsymbol{\theta},\boldsymbol{\phi}\big] + \mathrm{H}_{\check{p}\hat{p}}\big[\boldsymbol{Y}\,|\,\check{\boldsymbol{X}};\boldsymbol{\theta},\boldsymbol{\phi}\big] - \mathrm{H}_{\check{p}}\big[\check{\boldsymbol{X}}\,|\,\boldsymbol{Y};\boldsymbol{\phi}\big].
\end{aligned} \tag{6.30}$$

Now the right-hand side makes no reference to the model recognition distribution. And both sides are, of course, the free energy.

Let us consider some choices for the proxy recognition distribution. In "first-order general EM," discussed above, we use the Dirac delta, situated at the mode of the model recognition distribution. Clearly, $\mathrm{H}_{\check{p}}\big[\check{\boldsymbol{X}}\,|\,\boldsymbol{Y};\boldsymbol{\phi}\big] = 0$ under this choice—and the right-hand side of Eq. 6.30 becomes precisely the cost of communicating the observed data in the sender-receiver game, under hard assignment of data to the posterior mode! Thus we can say precisely how much we "overpay" in comparison to the mininal cost, $\mathrm{H}_{p\hat{p}}[\boldsymbol{Y};\boldsymbol{\theta}]$: the KL divergence between the Dirac delta and the true model recognition distribution.

An excess cost under the true model recognition distribution? Alternatively, suppose we *can* compute the model recognition distribution, and set $\check{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y}\big) = \hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$. Then the KL divergence vanishes, and the cost reduces to Eq. 6.29. How can we interpret the terms on the right-hand side in light of the sender-receiver game? We have seen that "recognizing" the latent states with Dirac deltas at the posterior modes corresponds to hard assignment in a sender-receiver game, e.g. of cluster identities. Likewise, the use of a non-deterministic (non-Dirac) recognition distribution corresponds to *soft* assignment of data to latent states. In particular, the first term in the cost, $\mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}};\boldsymbol{\theta},\boldsymbol{\phi}\big]$, corresponds to *sampling* from the model recognition distribution; in the GMM, to assigning the observation $\boldsymbol{y}$ to a cluster randomly, according to $\hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$.

For concreteness, we can imagine doing so according to the classic precedure: passing a sample from a uniform distribution through the inverse cumulative distribution function. This might seem like a terrible idea, since sampling appears to be "adding noise"—but we

shall see shortly how to recoup our loss. The second term in the cost, $\mathrm{H}_{\check{p}\hat{p}}\big[\boldsymbol{Y}|\check{\boldsymbol{X}};\boldsymbol{\theta},\boldsymbol{\phi}\big]$, accrues from encoding the "data misfits"—in the GMM, between the observation and the (randomly) chosen cluster mean.

It seems I will pay more than the optimum; to be precise, $\mathrm{H}_{\check{p}\hat{p}}\big[\check{\boldsymbol{X}}\,|\,\boldsymbol{Y};\boldsymbol{\theta},\boldsymbol{\phi}\big]$ bits more. This might be surprising, since we have assumed our model to be a good one, but perhaps it is the price of randomly assigning observations to latent states. Is there a way to recoup these lost bits?

Getting bits back. There is. For each "message" I send, the receiver can reconstruct $\boldsymbol{y}$ from the sample $\hat{\boldsymbol{x}}$ and the data misfit. But she can also recover the number used to sample $\hat{\boldsymbol{x}}$—by (in our concrete example) computing the cumulative probability of $\hat{\boldsymbol{x}}$. Furthermore, the uniformly distributed numbers need not have been random; they could be any data we also wish to send to the receiver. (We might have to transform them first to distribute them uniformly; this is an implementation detail.) The value of this "refund" will be precisely the cross entropy, $\mathrm{H}_{p\hat{p}}\Big[\hat{\boldsymbol{X}}\,|\,\boldsymbol{Y};\boldsymbol{\theta}\Big]$, of the recognition distribution, which we used to draw our samples, so upon getting these bits back, I will indeed have paid the minimal cost given by Eq. 6.29.

Thus, if the recognition distribution is maximally entropic—if, that is, each of the data are equally well assigned to each cloud of points—then the assignments I actually make communicate lots of information. At the other pole, if the model assigns data to clouds strictly, then this "extra channel" communicates no extra information. Most Gaussian mixture models will be somewhere between these extremes, in which case, under the stochastic assignment of latent variables, the receiver will get some bits back. But in every case the total cost will be $\mathrm{H}_{p\hat{p}}[\boldsymbol{Y};\boldsymbol{\theta}]$.

The encoding costs under proxy recognition distributions. If I *can't* compute the model recognition distribution, I can still use this "bits-back" argument, but the more general Eq. 6.30, rather than the special case of Eq. 6.29, applies. As when approximating the recognition distribution with a Dirac delta, the minimal extra cost will always be the relative entropy of the proxy and model recognition distributions, $\mathrm{D}_{\mathrm{KL}}\big\{\check{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y}\big)\big\|\hat{p}\big(\check{\boldsymbol{X}}\,\big|\,\boldsymbol{Y};\boldsymbol{\theta}\big)\big\}$. But unlike the case of the Dirac delta, the receiver will generally have to apply for bits back in order to reach this minimal cost. Is there an additional cost for sending the proxy recognition distribution, which the receiver needs to acquire these bits? No: she can simply train her own model on all the $(\hat{\boldsymbol{x}},\boldsymbol{y})$ pairs she has received or reconstructed.

Practical applications of the bits-back argument to compression. See Duda 2009, Havasi 2018, Kingma/Abbeel/Ho ICML 2019, Townshend ICRL 2019.

6.8 EFH-like models

Section 6.8.1 We return now to the undirected models of Chapter 3. Recall from our introduction to latent-variable density estimation above, Section 6.2, one of the motivations for introducing latent variables into our density-estimation problem: they can provide flexibility in modeling the observed data. Directed graphical models, and therefore expectation-maximization, are appropriate when we have furthermore a notion of how these latent variables are (marginally) distributed, and how the observations are conditionally distributed

given a setting of the latent variables; that is, when we have some idea of both $\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)$ and $\hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)$. In moving to models based on the architecture of the exponential-family harmonium (EFH), we trade an assumption about the form of $\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)$ for one about the form of $\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$.

In a way, however, this is a misleading description of our assumptions. For the directed generative models of the kind lately discussed, our construction of $\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)$ (and, perhaps, $\hat{p}\Big(\hat{\boldsymbol{Y}}\Big|\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)$) is often—though not always—motivated by some problem-specific semantic facts about the latent variables. In the (admittedly toy) problem in Fig. **??**, for example, the source is a Bernoulli distribution *because the hidden variable is taken to be the outcome of a coin flip*. By contrast, when adopting the harmonium as our latent-variable density estimator, we seldom have concrete ideas about the form of $\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$, and its selection is motivated mostly by mathematical convenience. Perhaps we require the hidden variables to have support in a certain set (e.g., the positive integers), but even this is uncommon. Most often, the individual elements of the vector of hidden units have no precise semantic content; rather, the vector as a whole represents no more than a kind of flexible set of parameters. After learning, they can be interpreted as detectors of "features" of the data, but there is still usually no *a priori* reason to assume that these features should be distributed according to $\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$. That is one reason why the RBM is so much more common than other exponential-family harmoniums: a binary vector will generally be just as suitable to represent features as, say, a vector of positive integers. In this, EFHs have a strong kinship with deterministic neural networks.

In any case, we do specify a form for $\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$, at the expense of freedom to choose $\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)$. Unfortunately, as discussed in Section 3.1, this method for making inference tractable (indeed, trivial), renders the joint distribution intractable. In particular, it is now the joint distribution, as well as the marginals over *both* $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$, that is specifiable only up to the normalizer ("partition function"). We therefore lose the ability to take (exact) expectations under these distributions, and are left with Monte Carlo techniques. But we have also undermined our ability to draw samples: In directed models, we simply sample from the source distribution (which we have) and then use this sample to sample from the emission probability (which we also have). In the harmonium, it is necessary instead to Gibbs sample—which can at least be performed layer-wise rather than unit-wise, but which nevertheless is painfully slow. So whereas for directed models, drawing one sample vector from the joint distribution requires one pass through all the variables, it may require (depending on the mixing rate of the Markov chain induced by the sampling procedure) scores to hundreds in the EFH.

Is the price of the trade worthwhile? Learning in the EFH and related models will require inference, naturally, but it will also require an expectation under the intractable model joint distribution. This expectation can be replaced with a sample average, but generating samples is, as we have just noted, equally difficult. However, after deriving the exact learning procedure (Section 6.8.1), we shall consider an alternative, approximate learning procedure, which obviates (or at least reduces) the need for prolonged Gibbs sampling (Section 6.8.2), allowing us to have our cake and eat it, too. The EFH also enjoys an even more remarkable property, which is an ability to be composed hierarchically, along with a guarantee that this

improves learning (more precisely, it lowers a bound). We shall explore such models, called "deep belief networks," in Section 6.8.3.

In what follows, we derive learning rules as generally as possible, usually for Boltzmann machines *tout court*, for which they can often be expressed elegantly, if not solved simply.

6.8.1 Minimizing relative entropy for exponential-family harmoniums

Relative-entropy minimization in energy-based models. We start with joint distributions of a rather general form known as a Boltzmann distribution[5]:

$$\begin{aligned} \hat{p}\Big(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big) &= \frac{1}{Z(\boldsymbol{\theta})} \exp\Big\{-E\Big(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}, \boldsymbol{\theta}\Big)\Big\}, \\ Z(\boldsymbol{\theta}) &= \sum_{\hat{\boldsymbol{x}}} \sum_{\hat{\boldsymbol{y}}} \exp\{-E(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \boldsymbol{\theta})\}. \end{aligned} \tag{6.31}$$

This is the generic form of distributions parameterized by undirected graphical models, although in this case we have made no assumptions about the underlying graph structure—indeed, we have not yet even distinguished the roles of $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$. Critically, without further constraints on the energy, the normalizer, $Z(\boldsymbol{\theta})$, is intractable for sufficiently large $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$, because the number of summands in Eq. 6.31 is exponential in $|\hat{\boldsymbol{X}}| + |\hat{\boldsymbol{Y}}|$. In some special cases of continuous random variables, the calculation of the normalizer may reduce to tractable integrals, but these cases are exceptional.

The goal, as usual, is to minimize the relative entropy of the distributions of observed data, $p(\boldsymbol{Y})$, and of model observations, $\hat{p}\Big(\hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big)$. The method is to follow its gradient in parameter space. Recalling from Eqs. 4.6 and 6.3:

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \mathrm{D}_{\mathrm{KL}}\{p(\boldsymbol{Y}) || \hat{p}(\boldsymbol{Y}; \boldsymbol{\theta})\} \approx \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \langle -\log \hat{p}(\boldsymbol{Y}; \boldsymbol{\theta}) \rangle_{\boldsymbol{Y}} = \left\langle -\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \log \hat{p}\Big(\hat{\boldsymbol{X}}, \boldsymbol{Y}; \boldsymbol{\theta}\Big) \right\rangle_{\hat{\boldsymbol{X}}, \boldsymbol{Y}}, \tag{6.32}$$

where the average is under the "hybrid" joint distribution $\hat{p}(\boldsymbol{X} | \boldsymbol{Y}; \boldsymbol{\theta}) p(\boldsymbol{Y})$. This is the point at which we opted, above, to use EM. The problem was that the average under a distribution that invokes parameters makes the right-hand side, in general, difficult to evaluate. Sometimes, indeed, $\hat{p}\Big(\hat{\boldsymbol{X}} \Big| \hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big)$ is not even available. Here it certainly is, and we make bold and try to apply Eq. 6.32 directly to the Boltzmann distribution, Eq. 6.31.

[5]The negative energy was originally referred to as the "harmony" [23], and omitting the negative sign does indeed frequently make the formulae prettier. But the base measure has already spoken for h, as entropy has for H—the capital η being indistinguishable from a capital h. So we work with energy, E, instead.

Continuing the progression:

$$\begin{aligned}
\left\langle -\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \log \hat{p}\Big(\hat{\boldsymbol{X}},\, \boldsymbol{Y}; \boldsymbol{\theta}\Big) \right\rangle_{\hat{\boldsymbol{X}}, \boldsymbol{Y}} &= \left\langle \frac{\mathrm{d}\log Z}{\mathrm{d}\boldsymbol{\theta}} + \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\hat{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= \frac{\mathrm{d}\log Z}{\mathrm{d}\boldsymbol{\theta}} + \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\hat{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= \frac{1}{Z} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \sum_{\hat{\boldsymbol{x}}} \sum_{\hat{\boldsymbol{y}}} \exp\{-E(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \boldsymbol{\theta})\} + \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\hat{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= -\sum_{\hat{\boldsymbol{x}}} \sum_{\hat{\boldsymbol{y}}} \frac{1}{Z} \exp\{-E(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \boldsymbol{\theta})\} \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} + \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\hat{\boldsymbol{X}}, \boldsymbol{Y}} \\
&= -\mathbb{E}_{\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}} \left[\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right] + \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\hat{\boldsymbol{X}}, \boldsymbol{Y}} \\
&\approx \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\hat{\boldsymbol{X}}, \boldsymbol{Y}} - \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}}
\end{aligned} \tag{6.33}$$

where in the last line we note (as always) the move from expectation to sample average, under the model joint, as an approximation. In words, the gradient is equal to the difference between averaging a quantity under one distribution as opposed to another. The quantity is the gradient of the energy. The distributions are the "hybrid joint distribution," constructed by combining the data distribution (over $\boldsymbol{Y}$) with the model's recognition distribution (over $\hat{\boldsymbol{X}}$); and the model distribution.

KL-minimization in the harmonium. Now we make an assumption about the roles of $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$. As we saw in Section 3.1, constraining the conditional distributions $\hat{p}\Big(\hat{\boldsymbol{Y}} \Big| \hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big)$ and $\hat{p}\Big(\hat{\boldsymbol{X}} \Big| \hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big)$ only to be in exponential families and to be self-consistent implies that they take the form

$$\begin{aligned}
\hat{p}\Big(\hat{\boldsymbol{X}} \Big| \hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big) &= h(\hat{\boldsymbol{X}}) \exp\Big\{ \Big(\boldsymbol{b} + \mathbf{W}_{\hat{y}\hat{x}} \boldsymbol{U}(\hat{\boldsymbol{Y}})\Big)^{\mathrm{T}} \boldsymbol{T}(\hat{\boldsymbol{X}}) - A(\boldsymbol{\eta}(\hat{\boldsymbol{Y}})) \Big\}, \\
\hat{p}\Big(\hat{\boldsymbol{Y}} \Big| \hat{\boldsymbol{X}}; \boldsymbol{\theta}\Big) &= k(\hat{\boldsymbol{Y}}) \exp\Big\{ \Big(\boldsymbol{c} + \mathbf{W}_{\hat{y}\hat{x}}^{\mathrm{T}} \boldsymbol{T}(\hat{\boldsymbol{X}})\Big)^{\mathrm{T}} \boldsymbol{U}(\hat{\boldsymbol{Y}}) - B(\boldsymbol{\zeta}(\hat{\boldsymbol{X}})) \Big\}.
\end{aligned} \tag{6.34}$$

In this case, the joint distribution will take the form of a Boltzmann distribution, Eq. 6.31, with energy

$$E(\boldsymbol{\theta}) = -\boldsymbol{c}^{\mathrm{T}} \boldsymbol{U}(\hat{\boldsymbol{Y}}) - \boldsymbol{b}^{\mathrm{T}} \boldsymbol{T}(\hat{\boldsymbol{X}}) - \boldsymbol{U}(\hat{\boldsymbol{Y}})^{\mathrm{T}} \mathbf{W}_{\hat{y}\hat{x}}^{\mathrm{T}} \boldsymbol{T}(\hat{\boldsymbol{X}}) - \log\Big(h(\hat{\boldsymbol{X}}) k(\hat{\boldsymbol{Y}})\Big). \tag{6.35}$$

The parameter gradients are therefore just the elegant

$$\frac{\mathrm{d}E}{\mathrm{d}\mathbf{W}_{\hat{y}\hat{x}}} = -\boldsymbol{T}\,\boldsymbol{U}^{\mathrm{T}} \qquad \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{b}} = -\boldsymbol{T} \qquad \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{c}^{\mathrm{T}}} = -\boldsymbol{U}^{\mathrm{T}}.$$

(See Section A.1 in the appendices for a refresher on derivatives with respect to matrices. For brevity, from here on, we omit the arguments of the sufficient statistics.) Making this

substitution into Eq. 6.33 and using Eq. 6.32 yields:

$$\begin{aligned}
\frac{\mathrm{dD_{KL}}\{p(\boldsymbol{Y})||\hat{p}(\boldsymbol{Y};\boldsymbol{\theta})\}}{\mathrm{d}\mathbf{W}_{\hat{y}\hat{x}}} &\approx \left\langle \boldsymbol{T}\,\boldsymbol{U}^{\mathrm{T}}\right\rangle_{\hat{\boldsymbol{X}},\hat{\boldsymbol{Y}}} - \left\langle \boldsymbol{T}\,\boldsymbol{U}^{\mathrm{T}}\right\rangle_{\hat{\boldsymbol{X}},\boldsymbol{Y}} \\
\frac{\mathrm{dD_{KL}}\{p(\boldsymbol{Y})||\hat{p}(\boldsymbol{Y};\boldsymbol{\theta})\}}{\mathrm{d}\boldsymbol{b}} &\approx \left\langle \boldsymbol{T}\right\rangle_{\hat{\boldsymbol{X}},\hat{\boldsymbol{Y}}} - \left\langle \boldsymbol{T}\right\rangle_{\hat{\boldsymbol{X}},\boldsymbol{Y}} \\
\frac{\mathrm{dD_{KL}}\{p(\boldsymbol{Y})||\hat{p}(\boldsymbol{Y};\boldsymbol{\theta})\}}{\mathrm{d}\boldsymbol{c}^{\mathrm{T}}} &\approx \left\langle \boldsymbol{U}^{\mathrm{T}}\right\rangle_{\hat{\boldsymbol{X}},\hat{\boldsymbol{Y}}} - \left\langle \boldsymbol{U}^{\mathrm{T}}\right\rangle_{\hat{\boldsymbol{X}},\boldsymbol{Y}}.
\end{aligned} \tag{6.36}$$

Evidently, learning is complete when the expected sufficient statistics of the model and data (or, more precisely, the model and the hybrid joint distribution) match. However, the elegance of these equations disguises a great difficulty, alluded to at the outset: The first average in each pair is under the joint distribution, which we can compute only up to the intractable normalizer. This certainly rules out setting these derivatives to zero and solving for the parameters in closed formed. Although we can use gradient descent instead, this doesn't relieve us of the need to evaluate, at least approximately, these averages. One obvious possibility is Gibbs sampling, and indeed the conditional distributions Eq. 6.34 are generally chosen so as to make block sampling possible—sample the entire vector $\hat{\boldsymbol{x}}$ given $\hat{\boldsymbol{y}}$ and the entire vector $\hat{\boldsymbol{y}}$ given $\hat{\boldsymbol{x}}$.

6.8.2 The method of contrastive divergence

We reiterate the major limitation to the learning algorithm just described. We can see from Eq. 6.36 that traveling the gradient of the KL divergence between data and model distributions requires expectations under the model joint or marginals (in the positive terms). Since this integral or sum usually cannot be computed analytically, the only obvious way to take it is via Gibbs sampling from the model. Although EFHs are designed to facilitate such a sampling procedure—the lack of intralayer connections makes it possible to sample all of the hidden variables conditioned on the visible ones, and vice versa—it will still generally take many steps of sampling to "burn in" the network, and then even more for subsequent thinning to get independent samples. And then there is a second problem: a procedure based on sampling will introduce variance into our estimate of the gradient; and furthermore, this variance depends on the current value of the parameters ($\boldsymbol{\theta}$). Hinton likens it to "a horizontal sheet of tin that is resonating in such a way that some parts have strong vertical oscillations and other parts are motionless. Sand scattered on the tin will accumulate in the motionless areas even though the time-averaged gradient is zero everywhere" [8].

Here is an alternative that appears, at first glance, to be very crude [9]. We recall the reason that Gibbs sampling requires a long burn-in period: our choice of initializing vector, being arbitrary, may have very low probability under the model. The burn-in travels down the Markov chain away from it towards the "thick" parts of the distribution we aim to sample. What if we had a better way to initialize the chain? In particular, suppose we initialize it at *samples from the data distribution*, and then take only a few steps along the Markov chain. Certainly, late in training, when the model distribution resembles the data distribution, this is a sensible procedure. But early in training, data samples are unlikely to have high probability under the model. Still, perhaps this "bias"—toward the regions of observation space with high probability under the data distribution—is a small price to pay for the reduction in variance that would have been accumulated through many steps of Gibbs sampling.

To try to make this more precise, we write down a mathematical expression for our approximation [8]. We replace the averaging under the model distribution in Eq. 6.33 (for Boltzmann machines generally; or, for EFHs in particular, in Eq. 6.36) with averaging under a distribution we call $p^1(\boldsymbol{X}^1, \boldsymbol{Y}^1; \boldsymbol{\theta})$. This is perhaps best understood in terms of sampling. Make four successive draws, from: first the data distribution, $p^0(\boldsymbol{Y}^{\boldsymbol{o}})$; then the model recognition distribution, $\hat{p}(\boldsymbol{X}^0 | \boldsymbol{Y}^{\boldsymbol{o}}; \boldsymbol{\theta})$; followed by the model emission, $\hat{p}(\boldsymbol{Y}^1 | \boldsymbol{X}^0; \boldsymbol{\theta})$; and finally from the recognition distribution again, $\hat{p}(\boldsymbol{X}^1 | \boldsymbol{Y}^1; \boldsymbol{\theta})$. The first two draws are frow $p^0(\boldsymbol{X}^0, \boldsymbol{Y}^0; \boldsymbol{\theta})$; the second two are from $p^1(\boldsymbol{X}^1, \boldsymbol{Y}^1; \boldsymbol{\theta})$.[6] Under this notational convention, the "hybrid joint distribution," $\hat{p}(\boldsymbol{X}^{\boldsymbol{o}} | \boldsymbol{Y}^{\boldsymbol{o}}; \boldsymbol{\theta}) p^0(\boldsymbol{Y}^{\boldsymbol{o}})$, would be called $p^0(\boldsymbol{X}^0, \boldsymbol{Y}^0; \boldsymbol{\theta})$: the distribution formed by "zero" steps of Gibbs sampling (more precisely, one half step). And likewise, taking n steps yields samples from $p^n(\boldsymbol{X}^n, \boldsymbol{Y}^n; \boldsymbol{\theta})$.

At the end of this Markov chain lies the model distribution; hence we can say $p^\infty(\boldsymbol{X}^{\boldsymbol{\infty}} \boldsymbol{Y}^{\boldsymbol{\infty}}) = \hat{p}(\boldsymbol{X}^{\boldsymbol{\infty}}, \boldsymbol{Y}^{\boldsymbol{\infty}}; \boldsymbol{\theta})$ [8]. In fact, for the remainder of this section, we shall use p^∞ for the model joint. Now substituting $p^1(\boldsymbol{X}^1, \boldsymbol{Y}^1; \boldsymbol{\theta})$ for $p^\infty(\boldsymbol{X}^{\boldsymbol{\infty}}, \boldsymbol{Y}^{\boldsymbol{\infty}}; \boldsymbol{\theta})$ in Eq. 6.33:

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} &\stackrel{?}{=} \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\boldsymbol{X}^0, \boldsymbol{Y}^{\boldsymbol{o}}} - \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\boldsymbol{X}^1, \boldsymbol{Y}^1} \\ &= \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} - \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\boldsymbol{X}^1, \boldsymbol{Y}^1 | \boldsymbol{X}^0, \boldsymbol{Y}^{\boldsymbol{o}}} \right\rangle_{\boldsymbol{X}^0, \boldsymbol{Y}^{\boldsymbol{o}}}, \end{aligned} \tag{6.37}$$

where in the last line we have emphasized that samples from $p^1(\boldsymbol{X}^1, \boldsymbol{Y}^1; \boldsymbol{\theta})$ would typically be nested inside the sampling from $p^0(\boldsymbol{X}^0, \boldsymbol{Y}^0; \boldsymbol{\theta})$. We have supposed that the learning rule in Eq. 6.37 is the gradient of some (unknown) loss function $\mathcal{L}$. This is clearly an attractive learning rule from the perspective of number of required samples, but will it produce good density estimators?

Contrastive divergence for energy-based models. From here we try to work backwards: what loss function might yield this gradient? Hinton proposes "contrastive divergence," i.e., the difference between the original KL divergence and a different one:

$$\mathcal{L}(\boldsymbol{\theta}) := \mathrm{D}_{\mathrm{KL}}\{p^0(\boldsymbol{Y}^{\boldsymbol{o}}) || p^\infty(\boldsymbol{Y}^{\boldsymbol{o}}; \boldsymbol{\theta})\} - \mathrm{D}_{\mathrm{KL}}\{p^n(\boldsymbol{Y}^n; \boldsymbol{\theta}) || p^\infty(\boldsymbol{Y}^n; \boldsymbol{\theta})\}. \tag{6.38}$$

This is an interesting quantity. First of all, it is positive. Why? Because the n-step distribution is "trapped" between p^0 and p^∞. That is, Gibbs sampling down the Markov chain brings distributions closer to p^∞, which will consequently diverge less from "later" than "earlier" distributions. So the second term is always less then the first, and their difference always positive. Second, the contrastive divergence is zero at precisely the same points as the first term—i.e., the classic KL penalty (as in Eq. 6.32). This can be seen by noting that the contrastive divergence can be zero only when the divergences are equal, i.e. when $p^n \approx p^0$. But this implies that p^0 is an equilibrium distribution for a Markov chain with the transition operator given by a single step of Gibbs sampling under the model. Further applications of this transition operator—additional steps of sampling from the model—cannot change this equilibrium distribution. So in this case, p^0 would be equal to p^m for all m, and both divergences would vanish.

[6] Note that to draw any sample from $p^1(\boldsymbol{X}^1, \boldsymbol{Y}^1; \boldsymbol{\theta})$ requires first drawing new samples $\boldsymbol{x}^0, \boldsymbol{y}^0$ from $p^0(\boldsymbol{X}^0, \boldsymbol{Y}^0; \boldsymbol{\theta})$; for any single sample $\boldsymbol{x}^0$, drawing from the model emission and then the recognition distributions amounts to drawing from the conditional distribution, $p^1(\boldsymbol{X}^1, \boldsymbol{Y}^1 | \boldsymbol{x}^0; \boldsymbol{\theta})$.

We might take this as a specific instance of a more general procedure: if the gradient of a function is difficult, replace it with a different function with the same minimum but an easier gradient. But *is* the gradient of the contrastive divergence easier to evaluate than the gradient of the KL divergence?

We have already worked out the gradient of the first KL divergence, Eqs. 6.32 and 6.33, but we break the computation into steps to reuse portions for the derivative of the second divergence. For brevity, we suppress the arguments:

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\big[\mathrm{D_{KL}}\{p^0||p^\infty\} - \mathrm{D_{KL}}\{p^n||p^\infty\}\big] \\
&= \frac{\partial \mathrm{D_{KL}}\{p^0||p^\infty\}}{\partial p^0}\frac{\mathrm{d}p^0}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \mathrm{D_{KL}}\{p^0||p^\infty\}}{\partial p^\infty}\frac{\mathrm{d}p^\infty}{\mathrm{d}\boldsymbol{\theta}} - \frac{\partial \mathrm{D_{KL}}\{p^n||p^\infty\}}{\partial p^n}\frac{\mathrm{d}p^n}{\mathrm{d}\boldsymbol{\theta}} - \frac{\partial \mathrm{D_{KL}}\{p^n||p^\infty\}}{\partial p^\infty}\frac{\mathrm{d}p^\infty}{\mathrm{d}\boldsymbol{\theta}} \\
&= 0 + \left(\left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\boldsymbol{X}^0,\boldsymbol{Y}^0} - \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\boldsymbol{X}^\infty,\boldsymbol{Y}^\infty}\right) - \frac{\partial \mathrm{D_{KL}}\{p^n||p^\infty\}}{\partial p^n}\frac{\mathrm{d}p^n}{\mathrm{d}\boldsymbol{\theta}} - \left(\left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\boldsymbol{X}^n,\boldsymbol{Y}^n} - \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\boldsymbol{X}^\infty,\boldsymbol{Y}^\infty}\right) \\
&= \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\boldsymbol{X}^0,\boldsymbol{Y}^0} - \left\langle \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} \right\rangle_{\boldsymbol{X}^n,\boldsymbol{Y}^n} - \frac{\partial \mathrm{D_{KL}}\{p^n||p^\infty\}}{\partial p^n}\frac{\mathrm{d}p^n}{\mathrm{d}\boldsymbol{\theta}}.
\end{aligned} \tag{6.39}$$

On the third line we have substituted for the fourth term by exploting the analogy with the second term (which we worked out in Eq. 6.33 above).

The final line is seemingly quite close to the desired gradient, Eq. 6.37. Is this close enough? Hinton reports that extensive simulation seems to indicate that this last term is small in practice, and that in any case its inclusion rarely changes the approximate version (Eq. 6.37) by more than 90 degrees; that is, neglecting it seldom makes the gradient point in the wrong direction [8]. And inefficiencies introduced by ignoring this term can perhaps be alleviated by choosing an n that balances the cost of these inaccurate gradients against the computational costs of protracted Gibbs sampling. After all, since the final term must disappear as the number of these "contrastive-divergence steps" approaches infinity, as long as there is no reason to suspect a discontinuity in the Markov chain, there will be some finite number n of CD steps that will shrink the nuisance term to an arbitrarily small size.

Contrastive divergence for the harmonium. The derivation in Eq. 6.39 is general to models based on the Boltzmann distribution. Applying the contrastive-divergence approximate learning rule, Eq. 6.37, to exponential-family harmoniums just means substituting $p^n(\boldsymbol{X}^n, \boldsymbol{Y}^n; \boldsymbol{\theta})$ for $\hat{p}\Big(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}; \boldsymbol{\theta}\Big)$ in Eq. 6.36. Here we write the one-step version, emphasizing the source of the samples of $p^1\big(\boldsymbol{X}^1, \boldsymbol{Y}^1; \boldsymbol{\theta}\big)$ by rearranging the averaging brackets as we did in Eq. 6.37. We also let the sufficient statistics be the vector random variables themselves, $\boldsymbol{U}(\hat{\boldsymbol{Y}}) = \hat{\boldsymbol{Y}}$, $\boldsymbol{T}(\hat{\boldsymbol{X}}) = \hat{\boldsymbol{X}}$ (what we will later have occasion to call "linear sufficient statistics"—see Chapter **??**), in order to facilitate a biological interpretation:

$$\begin{aligned}
\Delta \mathbf{W}_{\hat{y}\hat{x}} &\propto \left\langle \boldsymbol{X}^0 \boldsymbol{Y}^{0\mathrm{T}} - \left\langle \boldsymbol{X}^1 \boldsymbol{Y}^{1\mathrm{T}} \right\rangle_{\boldsymbol{X}^1,\boldsymbol{Y}^1|\boldsymbol{X}^0,\boldsymbol{Y}^0} \right\rangle_{\boldsymbol{X}^0,\boldsymbol{Y}^0}, \\
\Delta \boldsymbol{b} &\propto \left\langle \boldsymbol{X}^0 - \left\langle \boldsymbol{X}^1 \right\rangle_{\boldsymbol{X}^1|\boldsymbol{X}^0,\boldsymbol{Y}^0} \right\rangle_{\boldsymbol{X}^0,\boldsymbol{Y}^0}, \\
\Delta \boldsymbol{c} &\propto \left\langle \boldsymbol{Y}^0 - \left\langle \boldsymbol{Y}^1 \right\rangle_{\boldsymbol{Y}^1|\boldsymbol{X}^0,\boldsymbol{Y}^0} \right\rangle_{\boldsymbol{X}^0,\boldsymbol{Y}^0}.
\end{aligned} \tag{6.40}$$

Thought of as a neural network, we can say that each sample vector drives activity in the layer above, which reciprocally drives activity in the layer below, which drives activity in the layer above; after which "synaptic strengths" change proportional to pairwise correlations (average products) between the pre- and post-synaptic units, with the initial contributions being Hebbian and the final contributions being anti-Hebbian. For CD_n, $n > 1$, the reciprocal activity simply continues for longer before synaptic changes are made.

Some examples.

6.8.3 Learning in deep belief networks

We summarize our progress in learning in EFH-like models. Since we will shortly need to introduce subscripts to the random variables, we avoid clutter from here on by setting aside contrastive divergence and working out the results in terms of the original loss function (see Eq. 6.32)—allowing us to dispense with superscripts. Here, then, we return to using $\hat{p}$ rather than p^{∞} for the model distribution.

To recapitulate: the EFH represents, in some sense, a generative model more "agnostic" than the directed ones learned with EM: rather than committing to a source distribution of latent variables, $\hat{p}\Big(\hat{\boldsymbol{X}};\boldsymbol{\theta}\Big)$, it licenses a certain inference procedure to them, $\hat{p}\Big(\hat{\boldsymbol{X}}\,\Big|\,\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$. The price we pay for the ease of inference is difficulty in generating samples from, or taking expectations under, the unconditional (joint or marginal) model distributions. Computing the gradients for latent-variable density estimation (Eq. 6.33) requires just such operations, so the price may appear to be steep. However, an alternative procedure which requires very few samples, the method of contrastive divergence (Eq. 6.37), works well in practice. It can be justified either as a crude approximation to the original learning procedure, or as the approximate descent of an objective function slightly different from, but having the same minimum as, the original.

Supposing even that our learning procedure does minimize the KL divergence between data $p(\boldsymbol{Y})$ and model $\hat{p}\Big(\hat{\boldsymbol{Y}};\boldsymbol{\theta}\Big)$ distributions, that divergence may not be zero. The statistics of the data may be too rich to be representable by an EFH, or (say) one with a given number of hidden units. It would be nice to have a procedure for augmenting this trained EFH that is guaranteed to improve performance, i.e., to decrease KL divergence.

From incomplete cross entropy to free energy. Our derivation of just such a procedure begins with an idea: perhaps only certain parts of the trained, but insufficient, EFH want improvement. Specifically, we shall commit ourselves to using the generative distribution it has learned, $\hat{p}\Big(\hat{\boldsymbol{Y}}_0\Big|\hat{\boldsymbol{X}}_0;\boldsymbol{\theta}_0\Big)$, but try to replace the source distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}_0;\boldsymbol{\theta}_0\Big)$. (Here we have numbered the parameters—and the corresponding random variables—in anticipation of the introduction of more.) Of course, the whole point of the EFH is that the source distribution is learned only implicitly, so it may seem an infelicitous candidate for replacement. In particular, if we substitute a new source distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}_1;\boldsymbol{\theta}_1\Big)$, we shall seemingly be unable to avail ourselves of the EFH learning procedures—e.g., the contrastive-divergence parameter updates, Eq. 6.40. These rules require the recognition distribution, and although we have one consistent with $\hat{p}\Big(\hat{\boldsymbol{X}}_0;\boldsymbol{\theta}_0\Big)$ and $\hat{p}\Big(\hat{\boldsymbol{Y}}_0\Big|\hat{\boldsymbol{X}}_0;\boldsymbol{\theta}_0\Big)$, to wit $\hat{p}\Big(\hat{\boldsymbol{X}}_0\Big|\hat{\boldsymbol{Y}}_0;\boldsymbol{\theta}_0\Big)$, we do not have in hand one consistent with

$\hat{p}\Big(\hat{\boldsymbol{X}}_1;\boldsymbol{\theta}_1\Big)$ and $\hat{p}\Big(\hat{\boldsymbol{Y}}_0\Big|\hat{\boldsymbol{X}}_0;\boldsymbol{\theta}_0\Big)$—call it $\hat{p}\Big(\hat{\boldsymbol{X}}_1\Big|\hat{\boldsymbol{Y}}_0;\boldsymbol{\theta}_0,\boldsymbol{\theta}_1\Big)$. Nor is it clear how we should get one (although the reader is invited to think about this).

So we turn for inspiration to expectation-maximization (EM), the learning procedure for directed graphical models, which is in some sense what our problem has become, since we now have a parameterized generative distribution, $\hat{p}\Big(\hat{\boldsymbol{Y}}_0\Big|\hat{\boldsymbol{X}}_0;\boldsymbol{\theta}_0\Big)$, and source distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}_1;\boldsymbol{\theta}_1\Big)$, even if the latter has yet to be specified. Recall from Section 6.3 that our problem may be simplified if we trade descent in the cross entropy of the model under the data for descent in an upper bound, the "free energy." The slack in the bound is taken up by the KL divergence between a proxy recognition distribution and its true, but possibly uncomputable, counterpart (recall Eq. **??**). In the present case, let us select the true recognition distribution under the *original* EFH, i.e., $\hat{p}\Big(\hat{\boldsymbol{X}}_0\Big|\hat{\boldsymbol{Y}}_0;\boldsymbol{\theta}_0\Big)$, as the proxy. As we have lately noted, abandoning this model's source distribution presumably spoils the validity of its recognition distribution in the augmented model—that is, in general, $\hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}}_0;\boldsymbol{\theta}_0,\boldsymbol{\theta}_1\Big) \neq \hat{p}\Big(\hat{\boldsymbol{X}}\Big|\hat{\boldsymbol{Y}}_0;\boldsymbol{\theta}_0\Big)$. To emphasize that the right-hand side of the inequality (the original model recognition distribution) is a proxy for the left-hand side, we will subsequently refer to it as $\check{p}\big(\check{\boldsymbol{X}}_1\big|\,\boldsymbol{Y}_0\big)$.

Let us rewrite the free energy so as to separate out the pieces that are dependent on $\boldsymbol{\theta}_0$ and on $\boldsymbol{\theta}_1$:

$$\begin{aligned}
F(\{\boldsymbol{\theta}_0,\boldsymbol{\theta}_1\},\boldsymbol{\theta}_0) &= \mathrm{H}_{p\hat{p}}[\,\boldsymbol{Y}_0;\boldsymbol{\theta}_0,\boldsymbol{\theta}_1] + \mathrm{D}_{\mathrm{KL}}\big\{\check{p}\big(\check{\boldsymbol{X}}_1\big|\,\boldsymbol{Y}_0\big)\big|\big|\hat{p}\big(\check{\boldsymbol{X}}_1\big|\,\boldsymbol{Y}_0;\boldsymbol{\theta}_0,\boldsymbol{\theta}_1\big)\big\} \\
&= \left\langle -\log\hat{p}(\,\boldsymbol{Y}_0;\boldsymbol{\theta}_0,\boldsymbol{\theta}_1) + \sum_{\check{\boldsymbol{x}}_1}\hat{p}(\check{\boldsymbol{x}}_1|\,\boldsymbol{Y}_0;\boldsymbol{\theta}_0)[\log\check{p}(\check{\boldsymbol{x}}_1|\,\boldsymbol{Y}_0) - \log\hat{p}(\check{\boldsymbol{x}}_1|\,\boldsymbol{Y}_0;\boldsymbol{\theta}_0,\boldsymbol{\theta}_1)]\right\rangle_{\boldsymbol{Y}_0} \\
&= \big\langle\log\check{p}\big(\check{\boldsymbol{X}}_1\big|\,\boldsymbol{Y}_0\big) - \log\hat{p}\big(\check{\boldsymbol{X}}_1,\,\boldsymbol{Y}_0;\boldsymbol{\theta}_0,\boldsymbol{\theta}_1\big)\big\rangle_{\check{\boldsymbol{X}}_1,\boldsymbol{Y}_0} \\
&= \big\langle\log\check{p}\big(\check{\boldsymbol{X}}_1\big|\,\boldsymbol{Y}_0\big) - \log\hat{p}\big(\,\boldsymbol{Y}_0\big|\check{\boldsymbol{X}}_1;\boldsymbol{\theta}_0\big) - \log\hat{p}\big(\check{\boldsymbol{X}}_1;\boldsymbol{\theta}_1\big)\big\rangle_{\check{\boldsymbol{X}}_1,\boldsymbol{Y}_0}.
\end{aligned} \tag{6.41}$$

Therefore:

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}_1}F(\{\boldsymbol{\theta}_0,\boldsymbol{\theta}_1\},\boldsymbol{\theta}_0) = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}_1}\big\langle-\log\hat{p}\big(\check{\boldsymbol{X}}_1;\boldsymbol{\theta}_1\big)\big\rangle_{\check{\boldsymbol{X}}_1,\boldsymbol{Y}_0}. \tag{6.42}$$

Does this help?

Unrolling the EFH. Now we make a very clever observation [8]. Consider the "hybrid" graphical model in Fig. **??**. Samples from this model can be generated by, first, prolonged Gibbs sampling from the "inverted" EFH (notice the labels), followed by a single sample of the bottommost layer via the directed connections. But if the weight matrix defining these connections is the transpose of the EFH's weight matrix, then samples from this model are identical to samples from the visible layer of a standard EFH, Fig. **??**! Likewise, the recognition distributions from the models in Figs. **??** and **??**, $\check{p}\big(\check{\boldsymbol{X}}_1\big|\,\boldsymbol{Y}_0\big)$ and $\hat{p}\Big(\hat{\boldsymbol{X}}_1\Big|\hat{\boldsymbol{Y}}_0;\boldsymbol{\theta}_0,\boldsymbol{\theta}_0\Big)$, are identical.

Insert figure of (a) EFH, and (b) inverted EFH with one directed connection "hanging" off.

Suppose then we let the source distribution over $\hat{\boldsymbol{X}}_1$ in the model we seek to learn, $\hat{p}\Big(\hat{\boldsymbol{X}}_1;\boldsymbol{\theta}_1\Big)$, be the marginal distribution over $\hat{\boldsymbol{X}}_1$ in a second, inverted EFH, as in Fig. **??**. Moreover, let us initialize the parameters of this EFH at those of the first, fully trained EFH: $\boldsymbol{\theta}_1 \overset{\text{set}}{=} \boldsymbol{\theta}_0$. Then at this point in training, the KL divergence in the first line of Eq. 6.41 vanishes, and following the gradient in Eq. 6.42 decreases—at least at first—the cross entropy

of interest, $\mathrm{H}_{p\hat{p}}[\,\boldsymbol{Y}_0; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1]$. This is like a single M step of standard EM. There is no subsequent E step because there is no obvious way to bring the proxy recognition distribution, $\check{p}\big(\check{\boldsymbol{X}}_1 \big|\, \boldsymbol{Y}_0\big)$, back into agreement with the true recognition distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}_1 \Big|\, \hat{\boldsymbol{Y}}_0; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1\Big)$. We shall employ instead a different iterative improvement scheme (see below).

As soon as $\boldsymbol{\theta}_2$ is changed, the bound on this cross entropy can loosen: the KL divergence between proxy and model recognition distributions in the first line of Eq. 6.41 may increase. But, as in EM, so far from being problematic, this loosening would correspond to an even greater decrease in the cross entropy, since by hypothesis we are decreasing the free energy. That is, if this KL divergence grows, it does so at the expense of cross entropy; and if it doesn't (i.e., it stays at zero), then cross entropy still decreases. It is true that subsequent movement along this gradient can *decrease* the KL divergence between proxy and model recognition distributions, and therefore allow the cross entropy to increase, all while lowering the free energy. But the cross entropy can never increase past its value at the beginning of the changes of $\boldsymbol{\theta}_1$, since here the bound is tight, and all changes lower this bound. In fine, the network with a "rolled out" directed layer can never be worse (have higher cross entropy) than its point of departure, the original EFH of Fig. **??**. This is shown graphically in Fig. **??**.

Insert schematic plot of free entropy upper bounding cross entropy.

Recursive application of the procedure. Notice that descending the gradient in Eq. 6.42 decreases—unlike in EM—another marginal cross entropy. And this marginal distribution is treated as the "visible" layer of another EFH. Thus, the second EFH (defined by $\boldsymbol{\theta}_1$) is trained exactly the same way as the first (defined by $\boldsymbol{\theta}_0$), just on the empirical data as transformed by the first EFH's recognition distribution, $\sum_{\boldsymbol{y}_0} \check{p}\big(\check{\boldsymbol{X}}_1 \big| \boldsymbol{y}_0\big) p(\boldsymbol{y}_0) =: \check{p}\big(\check{\boldsymbol{X}}_1; \boldsymbol{\theta}_0\big)$, rather than the data in their raw form.

This suggests, what is true, that the entire procedure can be applied recursively. After training the second (inverted) EFH, the cross entropy in Eq. 6.42 may not be minimal, i.e., equal to the entropy of $\check{p}\big(\check{\boldsymbol{X}}_1; \boldsymbol{\theta}_0\big)$. Then a second directed layer can be "unrolled" from the EFH spool (Fig. **??**), and trained on a new "data distribution," consisting of the "data distribution," $\check{p}\big(\check{\boldsymbol{X}}_1; \boldsymbol{\theta}_0\big)$, used to train its predecessor, transformed by its predecessor's recognition distribution, $\check{p}\big(\check{\boldsymbol{Y}}_2 \big| \check{\boldsymbol{X}}_1; \boldsymbol{\theta}_1\big)$, to wit $\sum_{\check{\boldsymbol{x}}_1} \check{p}\big(\check{\boldsymbol{Y}}_2 \big| \check{\boldsymbol{x}}_1; \boldsymbol{\theta}_1\big) \check{p}(\check{\boldsymbol{x}}_1; \boldsymbol{\theta}_0) =: \check{p}\big(\check{\boldsymbol{Y}}_2; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1\big)$.

Can we guarantee that this will continue to decrease (or at least not increase) the model fit to the observed data? In fact we cannot, but we can guarantee something nearly as good: Either the model fits the data better, i.e. we reduce $\mathrm{H}_{p\hat{p}}[\,\boldsymbol{Y}_0; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2] = \big\langle -\log \hat{p}(\,\boldsymbol{Y}_0; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2)\big\rangle_{\boldsymbol{Y}_0}$; or the recognition distributions get closer to their proxies. And this guarantee extends to deeper layers. In fine, under this greedy, layer-wise training, either the resulting "deep belief network" becomes a better generative model, or it becomes a better recognition model.

To see this, we apply the argument, given in the previous section for the first "unrolled" EFH, to a second. Decoupling $\boldsymbol{\theta}_2$ from $\boldsymbol{\theta}_1$ and descending the gradient of the cross entropy $\big\langle -\log \hat{p}\big(\check{\boldsymbol{Y}}_2; \boldsymbol{\theta}_2\big)\big\rangle_{\check{\boldsymbol{Y}}_2}$—that is, improving the source distribution over $\hat{\boldsymbol{Y}}_2$—will at least initially decrease the cross entropy $\big\langle -\log \hat{p}\big(\check{\boldsymbol{X}}_1; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2\big)\big\rangle_{\check{\boldsymbol{X}}_1}$, by way of decreasing a free-energy upper bound which is initially tight (analogous to Eqs. 6.41 and 6.42). Now, $\hat{p}\Big(\hat{\boldsymbol{X}}_1; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2\Big)$ is likewise a source distribution over the subsequent layer, $\hat{\boldsymbol{Y}}_0$, so decreasing $\big\langle -\log \hat{p}\big(\check{\boldsymbol{X}}_1; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2\big)\big\rangle_{\check{\boldsymbol{X}}_1}$ decreases a second free-energy upper bound (Eqs. 6.41 and 6.42), albeit in this case *not one that is initially tight*. Therefore, decreasing this cross entropy *either* decreases the cross entropy of the next layer, $\big\langle -\log \hat{p}(\,\boldsymbol{Y}_0; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2)\big\rangle_{\boldsymbol{Y}_0}$, *or* it renders

Put in some figures for this!! Maybe you can use previous figs, but you should also have a (c) with three sets of parameters, so four layers.

this level's recognition distribution, $\hat{p}\Big(\hat{\boldsymbol{X}}_1 \Big| \hat{\boldsymbol{Y}}_0; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2\Big)$, a better match for the proxy recognition distribution, $\check{p}\big(\check{\boldsymbol{X}}_1 \big| \boldsymbol{Y}_0\big)$. And note that $\langle -\log \hat{p}(\boldsymbol{Y}_0; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2)\rangle_{\boldsymbol{Y}_0}$ is the cross entropy we actually care about: the deep belief network's fit to the observed data.

After the initial reduction of $\big\langle -\log \hat{p}\big(\check{\boldsymbol{Y}}_2; \boldsymbol{\theta}_2\big)\big\rangle_{\check{\boldsymbol{Y}}_2}$, i.e. the initial improvement of the source distribution over $\hat{\boldsymbol{Y}}_2$, the first free-energy bound can loosen, after which not all improvements in this source distribution translate into improvements in the source distribution over $\hat{\boldsymbol{X}}_1$. But (1) any other improvement comes from making $\hat{p}\Big(\hat{\boldsymbol{Y}}_2 \Big| \hat{\boldsymbol{X}}_1; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2\Big)$ more consistent with $\check{p}\big(\check{\boldsymbol{Y}}_2 \big| \check{\boldsymbol{X}}_1; \boldsymbol{\theta}_1\big)$; and (2) the source distribution over $\hat{\boldsymbol{X}}_1$, $\hat{p}\Big(\hat{\boldsymbol{X}}_1; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2\Big)$, can never get worse than it was before $\boldsymbol{\theta}_2$ was decoupled from $\boldsymbol{\theta}_1$. Extending the entire argument to deeper belief networks is straightforward.

Picturesquely, we can describe the whole procedure as follows. After training the first EFH, we freeze the way that "first-order features" give rise to data. But we need more flexibility in the source distribution of first-order features. So we train a new EFH to generate them. Our training procedure makes this second EFH more likely to generate features that *either* generate good data, $\hat{\boldsymbol{Y}}$, *or* are more consistent with the recognition model being used to generate the "training features." This is akin to an M step in EM (although we use stochastic gradient descent rather than solve explicitly for the minimum). When this learning process stalls out, we should like to tighten up the bound and begin again, as in the E step of EM, which would require somehow making the proxy recognition distribution more like the true one under our "hybrid" model. Instead, we simply repeat the procedure of replacing the source distribution—this time over "second-order features"—with another EFH. Although learning in this third EFH cannot be claimed to tighten the original bound, it does impose a tight bound on *this* bound: improvement in the generation of second-order features—the hidden variables for the first-order features—is guaranteed to improve first-order features, at least initially. Whether this improvement comes by way of improving data generation, or merely the bound on it, is not typically obvious.

The procedure is only guaranteed to keep lowering bounds (or bounds on bounds, etc.) if the new EFHs are introduced with their parameters initialized to the previous EFH's parameter values. In practice, however, this is almost always relaxed, because the new EFH is seldom even chosen to have the appropriate shape (same number of hidden units as the previous EFH's number of visible units). This allows for very general models; and although not guaranteed to keep improving performance, the introduction of deeper EFHs can be roughly justified along similar lines: it allows for more flexible source distributions over increasingly complex features.

notes on the assumption of a factorial posterior, complementary priors, etc.

Summary. We summarize the entire procedure for training deep belief networks with a pair of somewhat hairy definitions and a pair of learning rules. Pairs are required simply to maintain a connection to the notational distinction between latent and observed variables in the original EFH: conceptually, the training is the same at every layer.

$$\begin{aligned}
\check{p}\big(\check{\boldsymbol{Y}}_{m+1}; \boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_m\big) &:= \sum_{\check{\boldsymbol{x}}_m} \check{p}\big(\check{\boldsymbol{Y}}_{m+1} \big| \check{\boldsymbol{x}}_m; \boldsymbol{\theta}_m\big) \check{p}(\check{\boldsymbol{x}}_m; \boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_m) \qquad \text{for } m > 0 \text{ odd} \\
\check{p}\big(\check{\boldsymbol{X}}_{m+1}; \boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_m\big) &:= \sum_{\check{\boldsymbol{y}}_m} \check{p}\big(\check{\boldsymbol{X}}_{m+1} \big| \boldsymbol{y}_m\big) \check{p}(\check{\boldsymbol{y}}_m; \boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_m) \qquad \text{for } m \geq 0 \text{ even}
\end{aligned} \tag{6.43}$$

Then:

$$\Delta\boldsymbol{\theta}_m \propto -\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}_m}\big\langle -\log\hat{p}\big(\check{\boldsymbol{Y}}_m;\boldsymbol{\theta}_m\big)\big\rangle_{\check{\boldsymbol{Y}}_m} \qquad \text{for } m \geq 0 \text{ even}$$
$$\Delta\boldsymbol{\theta}_m \propto -\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}_m}\big\langle -\log\hat{p}\big(\check{\boldsymbol{X}}_m;\boldsymbol{\theta}_m\big)\big\rangle_{\check{\boldsymbol{X}}_m} \qquad \text{for } m > 0 \text{ odd.} \tag{6.44}$$

6.8.4 Dynamical models: the TRBM, RTRBM, and rEFH

Appendix A

Mathematical Appendix

A.1 Matrix Calculus

In the settings of machine learning and computational neuroscience, derivatives often appear in equations with matrices and vectors. Although it is always possible to re-express these equations in terms of sums of simpler derivatives, evaluating such expressions can be extremely tedious. It is therefore quite useful to have at hand rules for applying the derivatives directly to the matrices and vectors. The results are both easier to execute and more economically expressed. In defining these "matrix derivatives," however, some care is required to ensure that the usual formulations of the rules of scalar calculus—the chain rule, product rule, etc.—are preserved. We do that here.

Throughout, we treat vectors with a transpose (denoted $\boldsymbol{x}^{\mathrm{T}}$) as rows, and vectors without as columns.

A.1.1 Derivatives with respect to vectors

We conceptualize this fundamental operation as applying to vectors and yielding matrices. Application to scalars—or rather, scalar-valued functions—is then defined as a special case. Application to matrices (and higher-order tensors) is undefined.

The central idea in our definition is that the dimensions of the derivative must match the dimensions of the resulting matrix. In particular, we allow derivatives with respect to both column and row vectors; however:

1. In derivatives of a vector with respect to a vector, the two vectors must have opposite orientations; that is, we can take $\mathrm{d}\boldsymbol{y}/\,\mathrm{d}\boldsymbol{x}^{\mathrm{T}}$ and $\mathrm{d}\boldsymbol{y}^{\mathrm{T}}/\,\mathrm{d}\boldsymbol{x}$, but not $\mathrm{d}\boldsymbol{y}/\,\mathrm{d}\boldsymbol{x}$ or $\mathrm{d}\boldsymbol{y}^{\mathrm{T}}/\,\mathrm{d}\boldsymbol{x}^{\mathrm{T}}$. They are defined according to

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \mathbf{J}(\boldsymbol{y}) \qquad\qquad \frac{\mathrm{d}\boldsymbol{y}^{\mathrm{T}}}{\mathrm{d}\boldsymbol{x}} = \mathbf{J}^{\mathrm{T}}(\boldsymbol{y}),$$

   the Jacobian and its transpose.

Thus, the transformation of "shapes" behaves like an outer product: if $\boldsymbol{y}$ has length m and $\boldsymbol{x}$ has length n, then $\mathrm{d}\boldsymbol{y}/\,\mathrm{d}\boldsymbol{x}^{\mathrm{T}}$ is $m \times n$ and $\mathrm{d}\boldsymbol{y}^{\mathrm{T}}/\,\mathrm{d}\boldsymbol{x}$ is $n \times m$.

Several special cases warrant attention. Consider the *linear* vector-valued function $\boldsymbol{y} = \mathbf{A}\boldsymbol{x}$. Since $\boldsymbol{y}$ is a column, the derivative must be with respect to a row. In particular:

$$\frac{\mathrm{d}(\mathbf{A}\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \mathbf{A}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \mathbf{A}\mathbf{I} = \mathbf{A}.$$

Or again, consider the case where y is just a scalar function of $\boldsymbol{x}$. Rule 1 then says that $\mathrm{d}y/\,\mathrm{d}\boldsymbol{x}$ is a column-vector version of the gradient, and $\mathrm{d}y/\,\mathrm{d}\boldsymbol{x}^{\mathrm{T}}$ a row-vector version. When y is a *linear*, scalar function of $\boldsymbol{x}$, $\boldsymbol{c}\cdot\boldsymbol{x}$, the rule says that:

$$\frac{\mathrm{d}(\boldsymbol{c}\cdot\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \frac{\mathrm{d}(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{c})}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \frac{\mathrm{d}(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \boldsymbol{c}^{\mathrm{T}}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \boldsymbol{c}^{\mathrm{T}}\mathbf{I} = \boldsymbol{c}^{\mathrm{T}}$$
$$\frac{\mathrm{d}(\boldsymbol{c}\cdot\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}} = \frac{\mathrm{d}(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}} = \frac{\mathrm{d}(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{c})}{\mathrm{d}\boldsymbol{x}} = \frac{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}}{\mathrm{d}\boldsymbol{x}}\boldsymbol{c} = \mathbf{I}\boldsymbol{c} = \boldsymbol{c}.$$

The chain rule. Getting the chain rule right means making sure that the dimensions of the vectors and matrices generated by taking derivatives line up properly, which motivates the rule:

2. In the elements generated by the chain rule, all the numerators on the RHS must have the same orientation as the numerator on the LHS, and likewise for the denominators.

Rules 1 and 2, along with the requirement that inner matrix dimensions agree, ensure that the chain rule for a row-vector derivative is:

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}}\boldsymbol{z}(\boldsymbol{y}(\boldsymbol{x})) = \frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}\boldsymbol{y}^{\mathrm{T}}}\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}}.$$

This chain rule works just as well if z or y are scalars:

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}}\boldsymbol{z}(y(\boldsymbol{x})) = \frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}y}\frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} \qquad \text{(a matrix)}$$
$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}}z(\boldsymbol{y}(\boldsymbol{x})) = \frac{\mathrm{d}z}{\mathrm{d}\boldsymbol{y}^{\mathrm{T}}}\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} \qquad \text{(a row vector)}$$
$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}}z(y(\boldsymbol{x})) = \frac{\mathrm{d}z}{\mathrm{d}y}\frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} \qquad \text{(a row vector).}$$

We could write down the column-vector version by applying rule 2 while ensuring agreement between the inner matrix dimensions. Alternatively, we can apply rule 1 to the chain rule just derived for the row-vector derivative:

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}\boldsymbol{z}^{\mathrm{T}}(\boldsymbol{y}(\boldsymbol{x})) = \left(\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}\boldsymbol{y}^{\mathrm{T}}}\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}}\right)^{\mathrm{T}} = \frac{\mathrm{d}\boldsymbol{y}^{\mathrm{T}}}{\mathrm{d}\boldsymbol{x}}\frac{\mathrm{d}\boldsymbol{z}^{\mathrm{T}}}{\mathrm{d}\boldsymbol{y}}.$$

This is perhaps the less intuitive of the two chain rules, since it reverses the order in which the factors are usually written in scalar calculus.

The product rule. This motivates no additional matrix-calculus rules, but maintaining agreement among inner matrix dimensions does enforce a particular order. For example, let $y = \boldsymbol{u}(\boldsymbol{x})\cdot\boldsymbol{v}(\boldsymbol{x})$, the dot product of two vector-valued functions. Then the product rule must read:

$$\frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \frac{\mathrm{d}(\boldsymbol{v}^{\mathrm{T}}\boldsymbol{u})}{\mathrm{d}\boldsymbol{u}^{\mathrm{T}}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} + \frac{\mathrm{d}(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{v})}{\mathrm{d}\boldsymbol{v}^{\mathrm{T}}}\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \boldsymbol{v}^{\mathrm{T}}\mathbf{J}(\boldsymbol{u}) + \boldsymbol{u}^{\mathrm{T}}\mathbf{J}(\boldsymbol{v}).$$

The column-vector equivalent is easily derived by transposing the RHS. Neither, unfortunately, can be read as "the derivative of the first times the second, plus the first times the derivative of the second," as it is often taught in scalar calculus. It is easily remembered, nevertheless, by applying our rules 1 and 2, and checking inner matrix dimensions for agreement.

In the special case of a quadratic form, $y = \boldsymbol{x}^{\mathrm{T}}\mathbf{A}\boldsymbol{x}$, this reduces to:

$$\frac{\mathrm{d}(\boldsymbol{x}^{\mathrm{T}}\mathbf{A}\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \boldsymbol{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} + \boldsymbol{x}^{\mathrm{T}}\mathbf{A} = \boldsymbol{x}^{\mathrm{T}}(\mathbf{A}^{\mathrm{T}} + \mathbf{A}).$$

In the even more special case where A is symmetric, $\mathbf{A} = \mathbf{A}^{\mathrm{T}}$, this yields $2\boldsymbol{x}^{\mathrm{T}}\mathbf{A}$. Evidently, the column-vector equivalent is $2\mathbf{A}\boldsymbol{x}$.

A.1.2 Derivatives with respect to matrices

Scalar-valued functions. Given a matrix,

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix},$$

and a scalar-valued function y, we define:

$$\frac{\mathrm{d}y}{\mathrm{d}\mathbf{X}} = \begin{pmatrix} \frac{\mathrm{d}y}{\mathrm{d}x_{1,1}} & \frac{\mathrm{d}y}{\mathrm{d}x_{1,2}} & \cdots & \frac{\mathrm{d}y}{\mathrm{d}x_{1,n}} \\ \frac{\mathrm{d}y}{\mathrm{d}x_{2,1}} & \frac{\mathrm{d}y}{\mathrm{d}x_{2,2}} & \cdots & \frac{\mathrm{d}y}{\mathrm{d}x_{2,n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\mathrm{d}y}{\mathrm{d}x_{n,1}} & \frac{\mathrm{d}y}{\mathrm{d}x_{n,2}} & \cdots & \frac{\mathrm{d}y}{\mathrm{d}x_{n,n}} \end{pmatrix}.$$

This definition can be more easily applied if we translate it into the derivatives with respect to vectors introduced in the previous section. Giving names to the rows ($\bar{\boldsymbol{x}}_i^{\mathrm{T}}$) and columns ($\boldsymbol{x}_i$) of $\mathbf{X}$:

$$\mathbf{X} = \begin{pmatrix} \bar{\boldsymbol{x}}_1^{\mathrm{T}} \\ \bar{\boldsymbol{x}}_2^{\mathrm{T}} \\ \vdots \\ \bar{\boldsymbol{x}}_m^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{pmatrix},$$

we can write:

$$\frac{\mathrm{d}y}{\mathrm{d}\mathbf{X}} = \begin{pmatrix} \frac{\mathrm{d}y}{\mathrm{d}\bar{\boldsymbol{x}}_1{}^{\mathrm{T}}} \\ \frac{\mathrm{d}y}{\mathrm{d}\bar{\boldsymbol{x}}_2{}^{\mathrm{T}}} \\ \vdots \\ \frac{\mathrm{d}y}{\mathrm{d}\bar{\boldsymbol{x}}_m{}^{\mathrm{T}}} \end{pmatrix} = \begin{pmatrix} \frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{x}_1} & \frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{x}_2} & \cdots & \frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{x}_n} \end{pmatrix}. \tag{A.1}$$

This lets us more easily derive some common special cases. Consider the bilinear form $y = \boldsymbol{a}^{\mathrm{T}}\mathbf{X}\boldsymbol{b}$. The derivative with respect to the first row of $\mathbf{X}$ is:

$$\frac{\mathrm{d}(\boldsymbol{a}^{\mathrm{T}}\mathbf{X}\boldsymbol{b})}{\mathrm{d}\bar{\boldsymbol{x}}_1{}^{\mathrm{T}}} = \boldsymbol{a}^{\mathrm{T}} \begin{pmatrix} \frac{\mathrm{d}(\boldsymbol{b}^{\mathrm{T}}\bar{\boldsymbol{x}}_1)}{\mathrm{d}\bar{\boldsymbol{x}}_1{}^{\mathrm{T}}} \\ \frac{\mathrm{d}(\boldsymbol{b}^{\mathrm{T}}\bar{\boldsymbol{x}}_2)}{\mathrm{d}\bar{\boldsymbol{x}}_1{}^{\mathrm{T}}} \\ \vdots \\ \frac{\mathrm{d}(\boldsymbol{b}^{\mathrm{T}}\bar{\boldsymbol{x}}_m)}{\mathrm{d}\bar{\boldsymbol{x}}_1{}^{\mathrm{T}}} \end{pmatrix} = \boldsymbol{a}^{\mathrm{T}} \begin{pmatrix} \boldsymbol{b}^{\mathrm{T}} \\ \mathbf{0}^{\mathrm{T}} \\ \vdots \\ \mathbf{0}^{\mathrm{T}} \end{pmatrix} = a_1\boldsymbol{b}^{\mathrm{T}}$$

Stacking all m of these rows vertically as in Eq. A.1, we see that:

$$\frac{\mathrm{d}(\boldsymbol{a}^{\mathrm{T}}\mathbf{X}\boldsymbol{b})}{\mathrm{d}\mathbf{X}} = \begin{pmatrix} a_1\boldsymbol{b}^{\mathrm{T}} \\ a_2\boldsymbol{b}^{\mathrm{T}} \\ \vdots \\ a_m\boldsymbol{b}^{\mathrm{T}} \end{pmatrix} = \boldsymbol{a}\boldsymbol{b}^{\mathrm{T}}.$$

Alternatively, we might have used the column-gradient formulation:

$$\frac{\mathrm{d}(\boldsymbol{a}^{\mathrm{T}}\mathbf{X}\boldsymbol{b})}{\mathrm{d}\boldsymbol{x}_1} = \begin{pmatrix} \frac{\mathrm{d}(\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{a})}{\mathrm{d}\boldsymbol{x}_1} & \frac{\mathrm{d}(\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{a})}{\mathrm{d}\boldsymbol{x}_1} & \cdots & \frac{\mathrm{d}(\boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{a})}{\mathrm{d}\boldsymbol{x}_1} \end{pmatrix}\boldsymbol{b} = b_1\boldsymbol{a},$$

and then stacked these columns horizontally as in Eq. A.1:

$$\frac{\mathrm{d}(\boldsymbol{a}^{\mathrm{T}}\mathbf{X}\boldsymbol{b})}{\mathrm{d}\mathbf{X}} = \begin{pmatrix} b_1\boldsymbol{a} & b_2\boldsymbol{a} & \cdots & b_n\boldsymbol{a} \end{pmatrix} = \boldsymbol{a}\boldsymbol{b}^{\mathrm{T}}. \tag{A.2}$$

Or again, consider a case where $\mathbf{X}$ shows up in the other part of the bilinear form (in this case, a quadratic form):

$$y = (\mathbf{X}\boldsymbol{a} + \boldsymbol{b})^{\mathrm{T}}\mathbf{W}(\mathbf{X}\boldsymbol{a} + \boldsymbol{b}). \tag{A.3}$$

Then defining $\boldsymbol{z} := \mathbf{X}\boldsymbol{a} + \boldsymbol{b}$, and considering again just the first row of $\mathbf{X}$, we find:

$$\frac{\mathrm{d}y}{\mathrm{d}\bar{\boldsymbol{x}}_1{}^{\mathrm{T}}} = \frac{\mathrm{d}(\boldsymbol{z}^{\mathrm{T}}W\boldsymbol{z})}{\mathrm{d}\bar{\boldsymbol{x}}_1{}^{\mathrm{T}}} = (\mathbf{W}\boldsymbol{z})^{\mathrm{T}}\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}\bar{\boldsymbol{x}}_1{}^{\mathrm{T}}} + \boldsymbol{z}^{\mathrm{T}}\frac{\mathrm{d}(\mathbf{W}\boldsymbol{z})}{\mathrm{d}\bar{\boldsymbol{x}}_1{}^{\mathrm{T}}} = \boldsymbol{z}^{\mathrm{T}}(\mathbf{W}^{\mathrm{T}}+\mathbf{W})\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}\bar{\boldsymbol{x}}_1{}^{\mathrm{T}}} = \boldsymbol{z}^{\mathrm{T}}(\mathbf{W}^{\mathrm{T}}+\mathbf{W})\begin{pmatrix} \boldsymbol{a}^{\mathrm{T}} \\ \mathbf{0}^{\mathrm{T}} \\ \vdots \\ \mathbf{0}^{\mathrm{T}} \end{pmatrix} = v_1\boldsymbol{a}^{\mathrm{T}},$$

where $\boldsymbol{v} := (\mathbf{W} + \mathbf{W}^{\mathrm{T}})\boldsymbol{z}$, and v_1 is its first element. Stacking these rows vertically, as in Eq. A.1, yields:

$$\frac{\mathrm{d}y}{\mathrm{d}\mathbf{X}} = \boldsymbol{v}\boldsymbol{a}^{\mathrm{T}} = (\mathbf{W} + \mathbf{W}^{\mathrm{T}})(\mathbf{X}\boldsymbol{a} + \boldsymbol{b})\boldsymbol{a}^{\mathrm{T}}. \tag{A.4}$$

A common application of this derivative occurs when working with Gaussian functions, which can be written $e^{-y/2}$ for the y defined in Eq. A.3. In this case, the matrix $\mathbf{W}$ is symmetric, and the result simplifies further. More generally, Eq. A.3 occurs in quadratic penalties on the state in control problems, in which case X would be the state-transition matrix.

Matrix-valued functions. The derivative of a matrix-valued function with respect to a matrix is a tensor. These are cumbersome, so in a way our discussion of them is merely preliminary to what follows. Let x_{ij} be the $(i,j)^{\text{th}}$ entry of X. We consider a few simple matrix functions of X:

$$\frac{\mathrm{d}(\mathbf{A}\mathbf{X})}{\mathrm{d}x_{ij}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \underbrace{\boldsymbol{a}_i}_{j^{\text{th}}\text{ column}} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}$$

where $\boldsymbol{a}_i$ is the i^{th} column of $\mathbf{A}$. Transposes and derivatives commute, as usual, so the derivative of $\mathbf{X}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$ (e.g.) is just the transpose of the above. That means that

$$\frac{\mathrm{d}(\mathbf{X}\mathbf{A})}{\mathrm{d}x_{ij}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \underbrace{\tilde{\boldsymbol{a}}_j}_{i^{\text{th}}\text{ column}} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}^{\mathrm{T}},$$

with $\tilde{\boldsymbol{a}}_j^{\mathrm{T}}$ the j^{th} *row* of A. From the first we can also compute the slightly more complicated, but elegant:

$$\frac{\mathrm{d}(\mathbf{A}\mathbf{X}\mathbf{B}^{\mathrm{T}})}{\mathrm{d}x_{ij}} = \boldsymbol{a}_i \boldsymbol{b}_j^{\mathrm{T}},$$

with $\boldsymbol{b}_j$ the j^{th} column of $\mathbf{B}$. Finally, for a *square* matrix $\mathbf{A}$, we consider the even more complicated:

$$\frac{\mathrm{d}(\mathbf{X}\mathbf{A}\mathbf{X}^{\mathrm{T}})}{\mathrm{d}x_{ij}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \underbrace{\mathbf{X}\tilde{\boldsymbol{a}}_j}_{i^{\text{th}}\text{ column}} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}^{\mathrm{T}} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \underbrace{X\boldsymbol{a}_j}_{i^{\text{th}}\text{ column}} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}$$

For all of the above, the derivative with respect to the entire matrix $\mathbf{X}$ is just the collection of these matrices for all i and j.

Some applications of the chain rule to matrix derivatives. Now suppose we want to take a derivative with respect to $\mathbf{X}$ of the scalar-valued function $y(\mathbf{F}(\mathbf{X}))$, for various matrix-valued functions $\mathbf{F}(\cdot)$. We shall consider in particular those lately worked out. The chain rule here says that the $(i,j)^{\text{th}}$ element of this matrix is:

$$\frac{\mathrm{d}y(\mathbf{F}(\mathbf{X}))}{\mathrm{d}x_{ij}} = \sum_{kl} \frac{\mathrm{d}y}{\mathrm{d}F_{kl}} \frac{\mathrm{d}F_{kl}}{\mathrm{d}x_{ij}} = \mathbf{1}^{\mathrm{T}} \left(\frac{\mathrm{d}y}{\mathrm{d}\mathbf{F}} \circ \frac{\mathrm{d}\mathbf{F}}{\mathrm{d}x_{ij}} \right) \mathbf{1},$$

with $\mathbf{1}$ a vector of ones and $\circ$ the entry-wise (Hadamard) product. We now apply this equation to the results above:

$$\frac{\mathrm{d}y(\mathbf{A}\mathbf{X})}{\mathrm{d}x_{ij}} = \mathbf{1}^{\mathrm{T}} \left(\frac{\mathrm{d}y}{\mathrm{d}(\mathbf{A}\mathbf{X})} \circ \frac{\mathrm{d}(\mathbf{A}\mathbf{X})}{\mathrm{d}x_{ij}} \right) \mathbf{1} = \left\{ \frac{\mathrm{d}y}{\mathrm{d}(\mathbf{A}\mathbf{X})} \right\}_j^{\mathrm{T}} \boldsymbol{a}_i \implies \frac{\mathrm{d}y(\mathbf{A}\mathbf{X})}{\mathrm{d}\mathbf{X}} = \mathbf{A}^{\mathrm{T}} \frac{\mathrm{d}y}{\mathrm{d}(\mathbf{A}\mathbf{X})}, \tag{A.5}$$

$$\frac{\mathrm{d}y(\mathbf{X}\mathbf{A})}{\mathrm{d}x_{ij}} = \mathbf{1}^{\mathrm{T}} \left(\frac{\mathrm{d}y}{\mathrm{d}(\mathbf{X}\mathbf{A})} \circ \frac{\mathrm{d}(\mathbf{X}\mathbf{A})}{\mathrm{d}x_{ij}} \right) \mathbf{1} \implies \frac{\mathrm{d}y(\mathbf{X}\mathbf{A})}{\mathrm{d}\mathbf{X}} = \frac{\mathrm{d}y}{\mathrm{d}(\mathbf{X}\mathbf{A})} \mathbf{A}^{\mathrm{T}}, \tag{A.6}$$

$$\frac{\mathrm{d}y(\mathbf{A}\mathbf{X}\mathbf{B}^{\mathrm{T}})}{\mathrm{d}x_{ij}} = \mathbf{1}^{\mathrm{T}} \left(\frac{\mathrm{d}y}{\mathrm{d}(\mathbf{A}\mathbf{X}\mathbf{B}^{\mathrm{T}})} \circ \frac{\mathrm{d}(\mathbf{A}\mathbf{X}\mathbf{B}^{\mathrm{T}})}{\mathrm{d}x_{ij}} \right) \mathbf{1} \implies \frac{\mathrm{d}y(\mathbf{A}\mathbf{X}\mathbf{B}^{\mathrm{T}})}{\mathrm{d}\mathbf{X}} = \mathbf{A}^{\mathrm{T}} \frac{\mathrm{d}y}{\mathrm{d}(\mathbf{A}\mathbf{X}\mathbf{B}^{\mathrm{T}})} \mathbf{B}, \tag{A.7}$$

$$\frac{\mathrm{d}y(\mathbf{X}\mathbf{A}\mathbf{X}^{\mathrm{T}})}{\mathrm{d}x_{ij}} = \mathbf{1}^{\mathrm{T}} \left(\frac{\mathrm{d}y}{\mathrm{d}(\mathbf{X}\mathbf{A}\mathbf{X}^{\mathrm{T}})} \circ \frac{\mathrm{d}(\mathbf{X}\mathbf{A}\mathbf{X}^{\mathrm{T}})}{\mathrm{d}x_{ij}} \right) \mathbf{1} \implies \frac{\mathrm{d}y(\mathbf{X}\mathbf{A}\mathbf{X}^{\mathrm{T}})}{\mathrm{d}\mathbf{X}} = \frac{\mathrm{d}y}{\mathrm{d}(\mathbf{X}\mathbf{A}\mathbf{X}^{\mathrm{T}})} \mathbf{X}\mathbf{A}^{\mathrm{T}} + \frac{\mathrm{d}y}{\mathrm{d}(\mathbf{X}\mathbf{A}\mathbf{X}^{\mathrm{T}})}^{\mathrm{T}} \mathbf{X}\mathbf{A}. \tag{A.8}$$

A few special cases are interesting. Since the trace and the derivative are linear operators, they commute, and in particular

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{X}} \mathrm{tr}[\mathbf{X}] = \mathbf{I}.$$

Therefore, letting $y(\mathbf{X}) \overset{\text{set}}{=} \mathrm{tr}[\mathbf{X}]$ in the above equations, we have

$$\frac{\mathrm{d}\mathrm{tr}[\mathbf{A}\mathbf{X}]}{\mathrm{d}\mathbf{X}} = \frac{\mathrm{d}\mathrm{tr}[\mathbf{X}\mathbf{A}]}{\mathrm{d}\mathbf{X}} = \mathbf{A}^{\mathrm{T}}, \tag{A.9}$$

$$\frac{\mathrm{d}\mathrm{tr}\left[\mathbf{A}\mathbf{X}\mathbf{B}^{\mathrm{T}}\right]}{\mathrm{d}\mathbf{X}} = \mathbf{A}^{\mathrm{T}}\mathbf{B}, \tag{A.10}$$

$$\frac{\mathrm{d}\mathrm{tr}\left[\mathbf{X}\mathbf{A}\mathbf{X}^{\mathrm{T}}\right]}{\mathrm{d}\mathbf{X}} = \mathbf{X}\mathbf{A}^{\mathrm{T}} + \mathbf{X}\mathbf{A}. \tag{A.11}$$

A.1.3 More useful identities

Now that we have defined derivatives (of scalars and vectors) with respective to vectors and derivatives (of scalars) with respect to matrices, we can derive the following useful identities.

The derivative of the log-determinant. The trace and determinant of a matrix are related by a useful formula, derived through their respective relationships with the matrix's spectrum. Recall:

$$\begin{aligned}\mathrm{tr}[\mathbf{M}] &= \sum_i \lambda_i, \\ |\mathbf{M}| &= \prod_i \lambda_i.\end{aligned}$$

For each eigenvalue λ and associated eigenvector $\boldsymbol{v}$, $\mathbf{M}\boldsymbol{v} = \lambda\boldsymbol{v}$, so:

$$\begin{aligned}\exp\{\mathbf{M}\}\boldsymbol{v} &= \left(I + \mathbf{M} + \frac{\mathbf{M}^2}{2!} + \frac{\mathbf{M}^3}{3!} + \cdots\right)\boldsymbol{v} \\ &= \boldsymbol{v} + \lambda\boldsymbol{v} + \frac{\lambda^2}{2!}\boldsymbol{v} + \frac{\lambda^3}{3!}\boldsymbol{v} + \cdots \\ &= e^{\lambda}\boldsymbol{v}.\end{aligned}$$

Therefore, the eigenvectors of $\exp\{\mathbf{M}\}$ are the eigenvectors $\mathbf{M}$, and the eigenvalues of $\exp\{\mathbf{M}\}$ are the exponentiated eigenvalues of $\mathbf{M}$. Hence:

$$\exp\{\mathrm{tr}[\mathbf{M}]\} = \exp\left\{\sum_i \lambda_i\right\} = \prod_i \exp\{\lambda_i\} = |\exp\{\mathbf{M}\}|. \tag{A.12}$$

Therefore:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}x}\log|\mathbf{A}(x)| &= \frac{1}{|\mathbf{A}(x)|}\frac{\mathrm{d}}{\mathrm{d}x}|\mathbf{A}(x)| \\ \mathbf{M} := \log\mathbf{A} \implies &= \frac{1}{|\mathbf{A}(x)|}\frac{\mathrm{d}}{\mathrm{d}x}|\exp\{\mathbf{M}\}| \\ \text{Eq. A.12} \implies &= \frac{1}{|\mathbf{A}(x)|}\frac{\mathrm{d}}{\mathrm{d}x}\exp\{\mathrm{tr}[\mathbf{M}(x)]\} \\ &= \frac{1}{|\mathbf{A}(x)|}\exp\{\mathrm{tr}[\mathbf{M}(x)]\}\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{tr}[\mathbf{M}(x)] \\ &= \frac{1}{|\mathbf{A}(x)|}|\exp\{\mathbf{M}(x)\}|\mathrm{tr}\left[\frac{\mathrm{d}}{\mathrm{d}x}\mathbf{M}(x)\right] \\ &= \frac{1}{|\mathbf{A}(x)|}|\mathbf{A}(x)|\mathrm{tr}\left[\frac{\mathrm{d}}{\mathrm{d}x}\log\mathbf{A}(x)\right] \\ &= \mathrm{tr}\left[\mathbf{A}^{-1}\frac{\mathrm{d}\mathbf{A}}{\mathrm{d}x}\right].\end{aligned}$$

So far we have made use only of results from scalar calculus. (The derivative of the log of $\mathbf{A}(x)$ can be derived easily in terms of the Maclaurin series for the natural logarithm.)

ANOTHER EXAMPLE (fix me). Cf. the "trace trick," in e.g. IPGM. When $f = \log|\mathbf{A}|$ (i.e. the log of the determinant of $\mathbf{A}$),

$$\frac{\mathrm{d}\log|\mathbf{A}|)}{\mathrm{d}\mathbf{A}} = \mathbf{A}^{-\mathrm{T}}, \tag{A.13}$$

i.e. the inverse transpose. (This can be derived from the "interesting scalar case" below.) From this it follows easily that

$$\frac{\mathrm{d}|\mathbf{A}|}{\mathrm{d}\mathbf{A}} = |\mathbf{A}|\mathbf{A}^{-\mathrm{T}}. \tag{A.14}$$

A.2 Probability and Statistics

The exponential family and Generalized Linear Models (GLiMs)

Change of variables in probability densities

Distributions over linear combinations of random variables

- Have: $p(X, Y, O)$
- Define: $Z := aX + bY$

Hence:

$$p(Z|X, Y) = \delta(Z - (aX + bY)).$$

Recall that the so-called sifting property for a Dirac delta *of an arbitrary* $g(v)$ is:

$$\int_v f(v)\delta(g(v))\,\mathrm{d}v = \sum_i \frac{f(v_i)}{|g'(v_i)|},$$

where the v_i are the roots (assumed to be simple) of $g(v)$. Here we consider g as a function of Y: $g(Y) = Z - aX - bY$, which has the single root $(Z - aX)/b$. Thus:

$$\begin{aligned}\int_y p(Z|X, y)p(X, y, O)\,\mathrm{d}y &= \int_y \delta(Z - (aX + by))p(X, y, O)\,\mathrm{d}y \\ &= p(X, (Z - aX)/b, O)/b.\end{aligned}$$

But we also know that Z depends only on X and Y, and hence

$$p(Z|X, Y) = p(Z|X, Y, O).$$

Therefore

$$\begin{aligned}\int_y p(Z|X, y)p(X, y, O)\,\mathrm{d}y &= \int_y p(Z|X, y, O)p(X, y, O)\,\mathrm{d}y \\ &= \int_y p(X, y, Z, O)\,\mathrm{d}y \\ &= p(X, Z, O)\end{aligned}$$

Hence,

$$p_{xzo}(x, z, o) = p_{xyo}(x, (z - ax)/b, o)/b. \tag{A.15}$$

Things are more complicated, but more or less easily carried out, for Z defined as an arbitrary function of X and Y. Notice that for a simpler function, $a = b = 1$ ("coordinate transformation"), this reduces to:

$$p_{xzo}(x, z, o) = p_{xyo}(x, (z - x), o).$$

which says that to get the distribution over the sum Z (and everything else), we just substitute $(z - x)$ for y into the joint. By symmetry,

$$p_{yzo}(y, z, o) = p_{xyo}((z - y), y, o).$$

Thus to get, say, p_{zo}, one can either substitute $(Z - X)$ for Y in the joint and integrate out X, or substitute $(Z - Y)$ for X and integrate out Y.

The score function

The score is defined as the gradient of the log-likelihood (with respect to the parameters, $\boldsymbol{\theta}$), $\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \log \hat{p}\Big(\hat{\boldsymbol{Y}}_0; \boldsymbol{\theta}_0\Big)$. The mean of the score is zero:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{Y}}[\log \hat{p}(\boldsymbol{Y}; \boldsymbol{\theta}_0)] &= \int_{\boldsymbol{y}} \hat{p}(\boldsymbol{y}; \boldsymbol{\theta}_0) \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}_0} \log \hat{p}(\boldsymbol{y}; \boldsymbol{\theta}_0) \, \mathrm{d}\boldsymbol{y} \\
&= \int_{\boldsymbol{y}} \hat{p}(\boldsymbol{y}; \boldsymbol{\theta}_0) \frac{1}{\hat{p}(\boldsymbol{y}; \boldsymbol{\theta}_0)} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}_0} \hat{p}(\boldsymbol{y}; \boldsymbol{\theta}_0) \, \mathrm{d}\boldsymbol{y} \\
&= \int_{\boldsymbol{y}} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}_0} \hat{p}(\boldsymbol{y}; \boldsymbol{\theta}_0) \, \mathrm{d}\boldsymbol{y} \\
&= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}_0} \int_{\boldsymbol{y}} \hat{p}(\boldsymbol{y}; \boldsymbol{\theta}_0) \, \mathrm{d}\boldsymbol{y} \\
&= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}_0}(1) \\
&= 0.
\end{aligned}$$

The variance of the score is known as the Fisher information. Because its mean is zero, it is also the expected square of the score.

The Fisher information for exponential-family random variables

This turns out to take a simple form. For a (vector) random variable $\boldsymbol{Y}$ and "parameters" $\boldsymbol{\theta}$ (that may themselves be random variables):

$$p(\boldsymbol{Y}|\boldsymbol{\theta}) = p(\boldsymbol{Y}|\boldsymbol{\eta}) = h(\boldsymbol{Y}) \exp\left\{\boldsymbol{\eta}(\boldsymbol{\theta})^{\mathrm{T}} \boldsymbol{t}(\boldsymbol{Y}) - A(\boldsymbol{\eta}(\boldsymbol{\theta}))\right\},$$

the Fisher information is:

$$\begin{aligned} I(\boldsymbol{\theta}) &= -\mathbb{E}_{\boldsymbol{Y}|\boldsymbol{\theta}}\left[\frac{\partial^2}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^{\mathrm{T}}}\log p(\boldsymbol{Y}|\boldsymbol{\theta})\middle|\boldsymbol{\theta}\right] \\ &= -\mathbb{E}_{\boldsymbol{Y}|\boldsymbol{\theta}}\left[\frac{\partial^2}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^{\mathrm{T}}}\left[\boldsymbol{\eta}(\boldsymbol{\theta})^{\mathrm{T}}\boldsymbol{t}(\boldsymbol{Y}) - A(\boldsymbol{\eta}(\boldsymbol{\theta}))\right]\middle|\boldsymbol{\theta}\right] \\ &= -\mathbb{E}_{\boldsymbol{Y}|\boldsymbol{\theta}}\left[\sum_i \frac{\partial^2\eta_i}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^{\mathrm{T}}}t_i(\boldsymbol{Y}) - \frac{\partial\boldsymbol{\eta}^{\mathrm{T}}}{\partial\boldsymbol{\theta}}\frac{\partial^2 A}{\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}^{\mathrm{T}}}\frac{\partial\boldsymbol{\eta}}{\partial\boldsymbol{\theta}^{\mathrm{T}}} - \sum_i \frac{\partial^2\eta_i}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^{\mathrm{T}}}\frac{\partial A}{\partial\boldsymbol{\eta} i}\middle|\boldsymbol{\theta}\right] \\ &= \frac{\partial\boldsymbol{\eta}^{\mathrm{T}}}{\partial\boldsymbol{\theta}}\mathrm{Cov}_{\boldsymbol{Y}|\boldsymbol{\theta}}[\boldsymbol{t}(\boldsymbol{Y})|\boldsymbol{\theta}]\frac{\partial\boldsymbol{\eta}}{\partial\boldsymbol{\theta}^{\mathrm{T}}}, \end{aligned}$$

where in the last line we have used the fact that the derivatives of the log-normalizer are the cumulants of the sufficient statistics ($\boldsymbol{T}$) under the distribution. A perhaps more interesting equivalent can be derived by noting that:

$$\frac{\partial}{\partial\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{Y}|\boldsymbol{\theta}}[\boldsymbol{t}(\boldsymbol{Y})|\boldsymbol{\theta}] = \frac{\partial}{\partial\boldsymbol{\theta}}\frac{\partial A}{\partial\boldsymbol{\eta}^{\mathrm{T}}} = \frac{\partial^2 A}{\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}^{\mathrm{T}}}\frac{\partial\boldsymbol{\eta}}{\partial\boldsymbol{\theta}^{\mathrm{T}}} = \mathrm{Cov}_{\boldsymbol{Y}|\boldsymbol{\theta}}[\boldsymbol{t}(\boldsymbol{Y})|\boldsymbol{\theta}]\frac{\partial\boldsymbol{\eta}}{\partial\boldsymbol{\theta}^{\mathrm{T}}}.$$

Therefore,

$$\left(\frac{\partial}{\partial\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{Y}|\boldsymbol{\theta}}[\boldsymbol{t}(\boldsymbol{Y})|\boldsymbol{\theta}]\right)^{\mathrm{T}}\mathrm{Cov}_{\boldsymbol{Y}|\boldsymbol{\theta}}[\boldsymbol{t}(\boldsymbol{Y})|\boldsymbol{\theta}]^{-1}\left(\frac{\partial}{\partial\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{Y}|\boldsymbol{\theta}}[\boldsymbol{t}(\boldsymbol{Y})|\boldsymbol{\theta}]\right) = \frac{\partial\boldsymbol{\eta}^{\mathrm{T}}}{\partial\boldsymbol{\theta}}\mathrm{Cov}_{\boldsymbol{Y}|\boldsymbol{\theta}}[\boldsymbol{t}(\boldsymbol{Y})|\boldsymbol{\theta}]\frac{\partial\boldsymbol{\eta}}{\partial\boldsymbol{\theta}^{\mathrm{T}}} = I(\boldsymbol{\theta}). \tag{A.16}$$

Markov chains

Discrete random variables

[[[table]]]

Useful identities

Expectations of quadratic forms. Consider a vector random variable $\boldsymbol{X}$ with mean μ and covariance $\boldsymbol{\Sigma}$. We are interested in the expectation of a certain function of $\boldsymbol{X}$, namely $(\boldsymbol{b}-\mathbf{C}\boldsymbol{X})^{\mathrm{T}}\mathbf{A}(\boldsymbol{b}-\mathbf{C}\boldsymbol{X})$ This term can occur, for example, in the log probability of a Gaussian distribution about $\mathbf{C}\boldsymbol{X}$. To calculate the expectation, we define a new variable

$$\boldsymbol{Z} := \mathbf{A}^{1/2}(\boldsymbol{b}-\mathbf{C}\boldsymbol{X})$$

and then employ the cyclic-permutation property of the matrix-trace operator:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{X}}\Big[(\boldsymbol{b}-\mathbf{C}\boldsymbol{X})^{\mathrm{T}}\mathbf{A}(\boldsymbol{b}-\mathbf{C}\boldsymbol{X})\Big] &= \mathbb{E}_{\boldsymbol{Z}}\big[\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z}\big] \\ &= \mathbb{E}_{\boldsymbol{Z}}\big[\mathrm{tr}\big[\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z}\big]\big] \\ &= \mathbb{E}_{\boldsymbol{Z}}\big[\mathrm{tr}\big[\boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}}\big]\big] \\ &= \mathrm{tr}\big[\mathbb{E}_{\boldsymbol{Z}}\big[\boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}}\big]\big] \\ &= \mathrm{tr}\big[\mathrm{Cov}_{\boldsymbol{Z}}[\boldsymbol{Z}] + \mathbb{E}_{\boldsymbol{Z}}[\boldsymbol{Z}]\mathbb{E}_{\boldsymbol{Z}}\big[\boldsymbol{Z}^{\mathrm{T}}\big]\big] \\ &= \mathrm{tr}\Big[\mathbf{A}^{1/2}\mathbf{C}\Sigma\mathbf{C}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}/2} + \mathbf{A}^{1/2}(\boldsymbol{b}-\mathbf{C}\mu)(\boldsymbol{b}-\mathbf{C}\mu)^{\mathrm{T}}\mathbf{A}^{\mathrm{T}/2}\Big] \\ &= \mathrm{tr}\big[\mathbf{A}\mathbf{C}\Sigma\mathbf{C}^{\mathrm{T}}\big] + \mathrm{tr}\big[(\boldsymbol{b}-\mathbf{C}\mu)^{\mathrm{T}}\mathbf{A}(\boldsymbol{b}-\mathbf{C}\mu)\big] \\ &= \mathrm{tr}\big[\mathbf{A}\mathbf{C}\Sigma\mathbf{C}^{\mathrm{T}}\big] + (\boldsymbol{b}-\mathbf{C}\mu)^{\mathrm{T}}\mathbf{A}(\boldsymbol{b}-\mathbf{C}\mu) \end{aligned} \tag{A.17}$$

Hence, the expected value of the quadratic function of $\boldsymbol{X}$ is the quadratic function evaluated at the expected value of $\boldsymbol{X}$—plus a "correction" term arising from the covariance of $\boldsymbol{X}$.

A.3 Matrix Identities

For square, invertible matrices $\mathbf{A}$ and $\mathbf{B}$, the following are equivalent:

$$\mathbf{B}(\mathbf{A}+\mathbf{B})^{-1} \qquad \mathbf{I} - \mathbf{A}(\mathbf{A}+\mathbf{B})^{-1}$$
$$(\mathbf{I}+\mathbf{A}\mathbf{B}^{-1})^{-1} \qquad \mathbf{I} - (\mathbf{I}+\mathbf{B}\mathbf{A}^{-1})^{-1}$$
$$(\mathbf{A}^{-1}+\mathbf{B}^{-1})^{-1}\mathbf{A}^{-1} \qquad \mathbf{I} - (\mathbf{A}^{-1}+\mathbf{B}^{-1})^{-1}\mathbf{B}^{-1}$$

The proofs are by construction:

$$\begin{aligned}
\mathbf{B} = (\mathbf{A}+\mathbf{B}) - \mathbf{A} &\implies \mathbf{B}(\mathbf{A}+\mathbf{B})^{-1} = \mathbf{I} - \mathbf{A}(\mathbf{A}+\mathbf{B})^{-1} \\
\mathbf{B} = (\mathbf{A}+\mathbf{B}) - \mathbf{A} &\implies \mathbf{I} = (\mathbf{A}+\mathbf{B})\mathbf{B}^{-1} - \mathbf{A}\mathbf{B}^{-1} \\
&\implies (\mathbf{I}+\mathbf{A}\mathbf{B}^{-1})^{-1} = \mathbf{B}(\mathbf{A}+\mathbf{B})^{-1} \\
\mathbf{A}^{-1} = (\mathbf{A}^{-1}+\mathbf{B}^{-1}) - \mathbf{B}^{-1} &\implies (\mathbf{A}^{-1}+\mathbf{B}^{-1})^{-1}\mathbf{A}^{-1} = \mathbf{I} - (\mathbf{A}^{-1}+\mathbf{B}^{-1})^{-1}\mathbf{B}^{-1} \\
\mathbf{A}^{-1} = (\mathbf{A}^{-1}+\mathbf{B}^{-1}) - \mathbf{B}^{-1} &\implies \mathbf{B}\mathbf{A}^{-1} = \mathbf{B}(\mathbf{A}^{-1}+\mathbf{B}^{-1}) - \mathbf{I} \\
&\implies (\mathbf{I}+\mathbf{B}\mathbf{A}^{-1})^{-1} = (\mathbf{A}^{-1}+\mathbf{B}^{-1})^{-1}\mathbf{B}^{-1} \\
&\implies \mathbf{I} - (\mathbf{I}+\mathbf{B}\mathbf{A}^{-1})^{-1} = \mathbf{I} - (\mathbf{A}^{-1}+\mathbf{B}^{-1})^{-1}\mathbf{B}^{-1} \\
\mathbf{B}^{-1} = (\mathbf{A}^{-1}+\mathbf{B}^{-1}) - \mathbf{A}^{-1} &\implies \mathbf{A}\mathbf{B}^{-1} = \mathbf{A}(\mathbf{A}^{-1}+\mathbf{B}^{-1}) - \mathbf{I} \\
&\implies (\mathbf{I}+\mathbf{A}\mathbf{B}^{-1})^{-1} = (\mathbf{A}^{-1}+\mathbf{B}^{-1})^{-1}\mathbf{A}^{-1}
\end{aligned}$$

The Woodbury inversion lemma. For any "conformable" matrices $\mathbf{M}$ and $\mathbf{N}$, it is clearly the case that

$$\mathbf{M}(\mathbf{I}+\mathbf{N}\mathbf{M}) = (\mathbf{I}+\mathbf{M}\mathbf{N})\mathbf{M} \implies (\mathbf{I}+\mathbf{M}\mathbf{N})^{-1}\mathbf{M} = \mathbf{M}(\mathbf{I}+\mathbf{N}\mathbf{M})^{-1}.$$

We now find an alternative expression for $(\mathbf{I}+\mathbf{M}\mathbf{N})^{-1}$ using the above equation and the fact that $(\mathbf{I}+\mathbf{M}\mathbf{N})$ is its inverse:

$$\begin{aligned}
&\mathbf{I} = (\mathbf{I}+\mathbf{M}\mathbf{N})^{-1}(\mathbf{I}+\mathbf{M}\mathbf{N}) = (\mathbf{I}+\mathbf{M}\mathbf{N})^{-1} + (\mathbf{I}+\mathbf{M}\mathbf{N})^{-1}\mathbf{M}\mathbf{N} \\
\implies (\mathbf{I}+\mathbf{M}\mathbf{N})^{-1} &= \mathbf{I} - (\mathbf{I}+\mathbf{M}\mathbf{N})^{-1}\mathbf{M}\mathbf{N} \\
&= \mathbf{I} - \mathbf{M}(\mathbf{I}+\mathbf{N}\mathbf{M})^{-1}\mathbf{N}.
\end{aligned}$$

Finally, let $\mathbf{M} \overset{\text{set}}{=} \mathbf{A}^{-1}\mathbf{U}$ and $\mathbf{N} \overset{\text{set}}{=} \mathbf{C}\mathbf{V}$ in the above equation. Then

$$\begin{aligned}
&(\mathbf{I}+\mathbf{A}^{-1}\mathbf{U}\mathbf{C}\mathbf{V})^{-1} = \mathbf{I} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{I}+\mathbf{C}\mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{C}\mathbf{V} \\
\implies &(\mathbf{A}+\mathbf{U}\mathbf{C}\mathbf{V})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1}+\mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1}.
\end{aligned} \tag{A.18}$$

This final form is known as the Woodbury matrix-inversion lemma. Notice that we have assumed squareness and invertibility for $\mathbf{A}$ and $\mathbf{C}$, but not $\mathbf{U}$ or $\mathbf{V}$.

The special case in which $\mathbf{U}$ and $\mathbf{V}$ are vectors, $\boldsymbol{u}$ and $\boldsymbol{v}^{\mathrm{T}}$, and (without further loss of generality) $\mathbf{C} = 1$, is known as the Sherman-Morrison formula:

$$(\mathbf{A}+\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}})^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}\mathbf{A}^{-1}}{1+\boldsymbol{v}^{\mathrm{T}}\mathbf{A}^{-1}\boldsymbol{u}} \tag{A.19}$$

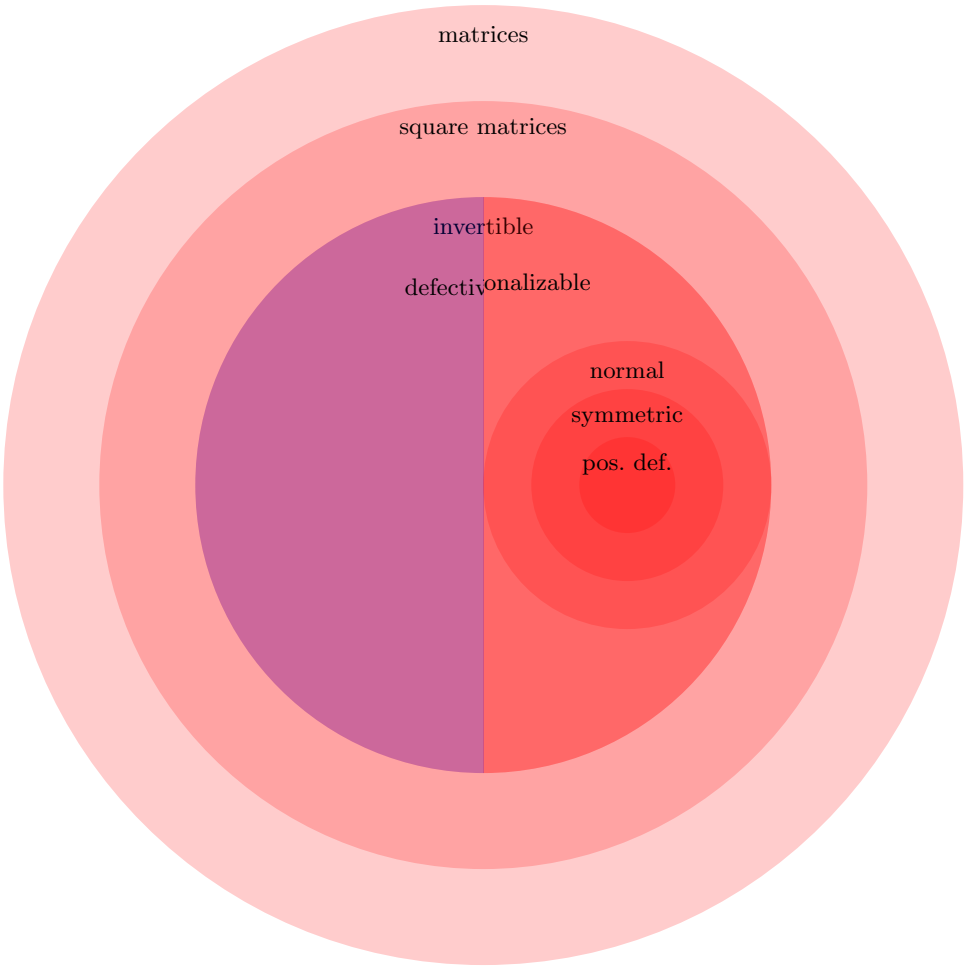


Figure A.1: **A taxonomy of matrices.**

Appendix B

A Review of Probabilistic Graphical Models

Bibliography

[1] Anthony J. Bell and Terrence J. Sejnowski. An Information-Maximization Approach to Blind Separation and Blind Deconvolution. *Neural Computation*, 7(6):1129–1159, nov 1995.

[2] J.F. Cardoso. Infomax and maximum likelihood for blind source separation. *Signal Processing Letters, IEEE*, 4(4):112–114, 1997.

[3] Peter Dayan, Geoffrey E. Hinton, R M Neal, and Richard S. Zemel. The Helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[4] R. A. Fisher. On the Mathematical Foundations of Theoretical Statistics. *Philosophical Transactions of the Royal Society A*, CCXXII:309–368, 1922.

[5] Per Christian Hansen, James G. Nagy, and Dianne P. O'Leary. *Deblurring Images: Matrices, Spectra, and Filtering*. SIAM, 2006.

[6] Michael Hartl. The Tau Manifesto. Accessed: 2022-05-09.

[7] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Westview Press, 1991.

[8] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14:1771–1800, 2002.

[9] Geoffrey E. Hinton and Andre Brown. Spiking Boltzmann Machines. *Advances in Neural Information Processing Systems 12: Proceedings of the 1999 Conference*, 12, 2000.

[10] Geoffrey E. Hinton and Zoubin Ghahramani. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 352(1358):1177, 1997.

[11] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, Minimum Description Length and Helmholtz Free Energy. In *Advances in Neural Information Processing Systems 6: Proceedings of the 1993 Conference*, pages 3–10, 1994.

[12] E.T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, ebook edition, 2003.

[13] Michael I. Jordan. Why the logistic function? A tutorial discussion on probabilities and neural networks. Technical report, 1995.

[14] Michael S. Lewicki and Bruno A. Olshausen. Probabilistic framework for the adaptation and comparison of image codes. *J. Opt. Soc. Am.*, 16(7):1587–1601, 1999.

[15] Michael S. Lewicki and Terrence J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2):337–65, feb 2000.

[16] Michael S. Lewicki and Terrence J Sejnowski. Learning Overcomplete Representations. *Neural Computation*, 12:337–365, 2000.

[17] Radford M. Neal and Geoffrey E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, 1998.

[18] Bruno A. Olshausen and DJ Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[19] Manfred Opper and Cedric Archambeau. The Variational Gaussian Approximation Revisited. *Neural Computation*, 21:786–792, 2009.

[20] L . R . Pericchi and A . F . M . Smith. Exact and Approximate Posterior Moments for a Normal Location Parameter. *Journal of the Royal Statistical Society*, 54(3):793–804, 1992.

[21] W.V.O. Quine. Two Dogmas of Empiricism. *The Philosophical Review*, 60:20–43, 1951.

[22] Eero P. Simoncelli and Bruno A. Olshausen. Natural Image Statistics and Neural Representations. *Annual Review of Neuroscience*, 24:1193–216, 2001.

[23] P Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, chapter 6, pages 194–281. MIT Press, 1986.

[24] Richard M Soland. Bayesian Analysis of the Weibull Process With Unknown Scale and Shape Parameters. *IEEE Transactions on Reliability*, R-18(4):181–184, 1969.

[25] Ilya Sutskever, Geoffrey E. Hinton, and Graham W Taylor. The Recurrent Temporal Restricted Boltzmann Machine. In *Advances in Neural Information Processing Systems 21: Proceedings of the 2008 Conference*, pages 1–8, 2009.

[26] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions*. Wiley, 1985.

[27] Max Welling, Michal Rosen-Zvi, and Geoffrey E. Hinton. Exponential Family Harmoniums with an Application to Information Retrieval. In *Advances in Neural Information Processing Systems 17: Proceedings of the 2004 Conference*, pages 1481–1488., 2005.