10 Representational Learning

10.1 Introduction

The response selectivities of individual neurons, and the way they are distributed across neuronal populations, define how sensory information is represented by neural activity. Sensory information is typically represented in multiple brain regions, the visual system being a prime example, with the nature of the representation shifting progressively along the sensory pathway. In previous chapters, we discussed how such representations can be generated by neural circuitry and developed by activity-dependent plasticity. In this chapter, we study neural representations from a computational perspective, asking what goals are served by particular representations, and how appropriate representations might be developed on the basis of input statistics.

re-representation

Constructing new representations of, or re-representing, sensory input is important because sensory receptors often deliver information in a form that is unsuitable for higher-level cognitive tasks. For example, roughly 10^8 photoreceptors provide a pixelated description of the images that appear on our retinas. A list of the membrane potentials of each of these photoreceptors provides a bulky and awkward representation of the visual world, from which it would be difficult to identify directly the face of a friend or a familiar object. Instead, the information provided by photoreceptor outputs is processed in a series of stages involving increasingly sophisticated representations of the visual world. In this chapter, we consider how these more complex and useful representations can be constructed.

generative model

The key to building useful representations for vision lies in determining the structure of visual images and capturing the constraints imposed on images by the natural world. The set of possible pixelated activities arising from natural scenes is richly structured and highly constrained, because images are not generated randomly, but arise from well-defined objects, such as rocks, trees, and people. We call these objects the "causes" of the images. In representational learning, we seek to identify causes by analyzing the statistical structure of visual images and building a model, called the generative model, that is able to reproduce this structure. Iden-

recognition model

tification of the causes of particular images (e.g., object recognition) is performed by a second model, called the recognition model, that is constructed on the basis of the generative model. This procedure is analogous to analyzing an experiment by building a model of the processes thought to underly it, and using the model as a basis for extracting interesting features from the accumulated experimental data.

input vector $\mathbf{u}$
cause v
hidden or latent variable

We follow the convention of previous chapters and use the variables $\mathbf{u}$ and v to represent the input and output of the models we consider. The input vector $\mathbf{u}$ represents the data that we wish to analyze in terms of underlying causes. The output v, which is a variable that characterizes those causes, is sometimes called the hidden or latent variable, but we call it the "cause". In some models, causes may be associated with a vector $\mathbf{v}$, rather than a single number v.

recognition
deterministic recognition
probabilistic recognition

In terms of these variables, the ultimate goal of the models we consider is recognition, in which the model tells us something about the causes v underlying a particular input $\mathbf{u}$. Recognition can be either deterministic or probabilistic. In a model with deterministic recognition, the output $v(\mathbf{u})$ is the model's estimate of the cause underlying input $\mathbf{u}$. In probabilistic recognition, the model estimates the probability that different values of v are associated with input $\mathbf{u}$. In either case, the output is taken as the model's re-representation of the input.

heuristics

We consider models that infer causes in an unsupervised manner. This means that the existence and identity of any underlying causes must be deduced solely from two sources of information. One is the set of general assumptions and guesses, collectively known as heuristics, concerning the nature of the input data and the causes that might underly them. These heuristics determined the general form of the generative model. The other source of information is the statistical structure of the input data. In the absence of supervisory information or even reinforcement, causes are judged by their ability to explain and reproduce the statistical structure of the inputs they are designed to represent. The process is analogous to judging the validity of a model by comparing simulated data generated by it with the results of a real experiment. The basic idea is to use assumed causes to generate synthetic input data from a generative model. The statistical structure of the synthetic data is then compared with that of the real input data, and the parameters of the generative model are adjusted until the two are as similar as possible. If the final statistical match is good, the causes are judged trustworthy, and the model can be used as a basis for recognition.

Representational learning is a large and complex subject with a terminology and methodology that may be unfamiliar to many readers. Section 10.1 follows two illustrative examples to provide a general introduction. This should give the reader a basic idea of what representational learning attempts to achieve and how it works. Section 10.2 covers more technical aspects of the approach, and 10.3 surveys a number of examples.

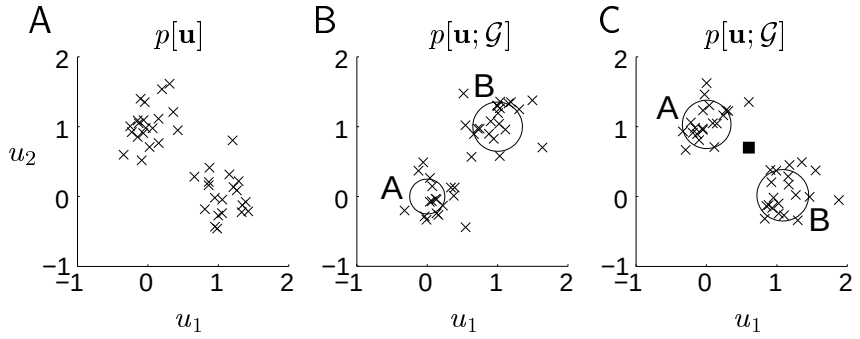

Figure 10.1 Clustering. (A) Input data points drawn from the distribution $p[\mathbf{u}]$ are indicated by the crosses. (B) Initialization for a generative model. The means and twice the standard deviations of the two Gaussians are indicated by the locations and radii of the circles. The crosses show synthetic data, which are samples from the distribution $p[\mathbf{u}; \mathcal{G}]$ of the generative model. (C) Means, standard deviations, and synthetic data points generated by the optimal generative model. The square indicates a new input point that is assigned to either cluster A or cluster B with probabilities computed from the recognition model.

Causal Models

Figure 10.1A provides a simple example of structured data that suggests underlying causes. In this case, the input takes the form of a two-component vector, $\mathbf{u} = (u_1, u_2)$. A collection of sample inputs that we wish to represent in terms of underlying causes is indicated by the 40 crosses in figure 10.1A. These inputs are drawn from a probability density $p[\mathbf{u}]$ that we call the input distribution. Clearly, there are two clusters of points in figure 10.1A, one centered near $(0, 1)$ and the other near $(1, 0)$.

input distribution $p[\mathbf{u}]$

Many processes can generate clustered data. For example, u_1 and u_2 might be characterizations of the voltage recorded on an extracellular electrode in response to an action potential. Interpreted in this way, these data suggest that we are looking at spikes produced by two neurons (called A and B), which are the underlying causes of the two clusters seen in figure 10.1A. A more compact and causal description of the data can be provided by a single output variable v that takes the value A or B for each data point, representing which of the two neurons was responsible for a particular action potential. Directly reading the output of such a model would be an example of deterministic recognition, with $v(\mathbf{u})$ = A or B providing the model's estimate of which neuron produced the spike associated with input $\mathbf{u}$. We consider, instead, a model with probabilistic recognition that estimates the probability that the spike with input data $\mathbf{u}$ was generated by either neuron A or neuron B.

In this example, we assume from the start that there are two possible, mutually exclusive causes for the data points, the two neurons A and B. By making this assumption, which is part of the heuristics underlying the generative model, we avoid the problem of identifying the number of pos-

sible causes (i.e., the number of clusters). Probabilistic methods can be used to make statistical inferences about the number of clusters in a data set, but they lie beyond the scope of this text.

Generative Models

To illustrate the concept of a generative model, we construct one (called a mixture of Gaussians model) for the data in figure 10.1A. The general form of the model is determined by the heuristics, assumptions about the nature of the causes and the way they generate inputs. However, the model has parameters that can be adjusted to fit the actual data that are observed. We begin by introducing parameters γ_A and γ_B that represent the proportions (also known as mixing proportions) of action potentials generated by each of the neurons. These might account for the fact that one of the neurons has a higher firing rate than the other, for example. The parameter γ_υ, with $\upsilon = \text{A}$ or B, specifies the probability $P[\upsilon; \mathcal{G}]$ that a given spike was generated by neuron υ in the absence of any knowledge about the input $\mathbf{u}$ associated with that spike. $P[\upsilon; \mathcal{G}] = \gamma_\upsilon$ is called the prior distribution over causes. The symbol $\mathcal{G}$ stands for all the parameters used to characterize the generative model. At this point, $\mathcal{G}$ consists of the two parameters γ_A and γ_B, but more parameters will be added as we proceed. We start by assigning γ_A and γ_B random values consistent with the constraint that they must sum to 1.

mixing proportions

prior $P[\upsilon; \mathcal{G}]$

parameters $\mathcal{G}$

To continue the construction of the generative model, we need to assume something about the distribution of $\mathbf{u}$ values arising from the action potentials generated by each neuron. An examination of figure 10.1A suggests that Gaussian distributions (with the same variance in both dimensions) might be appropriate. The probability density of $\mathbf{u}$ values given that neuron υ fired is $p[\mathbf{u}|\upsilon; \mathcal{G}]$. This is set to a Gaussian distribution with a mean and variance that, initially, we guess. The parameter list $\mathcal{G}$ now contains the prior probabilities, γ_A and γ_B, and the means and variances of the Gaussian distributions over $\mathbf{u}$ for $\upsilon = \text{A}$ and B, which we label $\mathbf{g}_\upsilon$ and Σ_υ, respectively. Note that Σ_υ is used to denote the variance of cluster υ, not its standard deviation, and also that each cluster is characterized by a single variance because we consider only circularly symmetric Gaussian distributions.

generative distribution $p[\mathbf{u}|\upsilon; \mathcal{G}]$

Figure 10.1B shows synthetic data points (crosses) generated by this model. To create each point, we set $\upsilon = \text{A}$ with probability $P[\upsilon = \text{A}; \mathcal{G}]$ (or otherwise set $\upsilon = \text{B}$) and then generated a point $\mathbf{u}$ randomly from the distribution $p[\mathbf{u}|\upsilon; \mathcal{G}]$. This generative model clearly has the capacity to create a data distribution with two clusters, similar to the one in figure 10.1A. However, the values of the parameters $\mathcal{G}$ used in figure 10.1B are obviously inappropriate. They must be adjusted by a learning procedure that matches, as accurately as possible, the distribution of synthetic data points in figure 10.1B to the actual input distribution in figure 10.1A. We describe how this is done in a later section. After optimization, as seen in

figure 10.1C, synthetic data points generated by the model (crosses) overlap well with the actual data points seen in figure 10.1A.

The distribution of synthetic data points in figures 10.1B and 10.1C is described by the probability density $p[\mathbf{u}; \mathcal{G}]$ that the generative model synthesizes an input with the value $\mathbf{u}$. This can be computed from the conditional density $p[\mathbf{u}|v; \mathcal{G}]$ and the prior distribution $P[v; \mathcal{G}]$ that define the generative model,

marginal distribution $p[\mathbf{u}; \mathcal{G}]$

$$p[\mathbf{u}; \mathcal{G}] = \sum_v p[\mathbf{u}|v; \mathcal{G}] P[v; \mathcal{G}] . \tag{10.1}$$

The process of summing over all causes is called marginalization, and $p[\mathbf{u}; \mathcal{G}]$ is called the marginal distribution over $\mathbf{u}$. As in chapter 8, we use the additional argument $\mathcal{G}$ to distinguish the distribution of synthetic inputs produced by the generative model, $p[\mathbf{u}; \mathcal{G}]$, from the distribution of actual inputs, $p[\mathbf{u}]$. The process of adjusting the parameters $\mathcal{G}$ to make the distributions of synthetic and real input data points match corresponds to making the marginal distribution $p[\mathbf{u}; \mathcal{G}]$ approximate, as closely as possible, the distribution $p[\mathbf{u}]$ from which the input data points are drawn.

In a later section, we make use of an addition probability distribution associated with the generative model, the joint probability distribution over both causes and inputs, define by

joint distribution $p[v, \mathbf{u}; \mathcal{G}]$

$$p[v, \mathbf{u}; \mathcal{G}] = p[\mathbf{u}|v; \mathcal{G}] P[v; \mathcal{G}] . \tag{10.2}$$

This describes the probability of cause v and input $\mathbf{u}$ both being produced by the generative model.

As mentioned previously, the choice of a particular structure for a generative model reflects our notions and prejudices (i.e., our heuristics) concerning the nature of the causes that underlie input data. Usually, the heuristics consist of biases toward certain types of representations, which are imposed through the choice of the prior distribution $p[v; \mathcal{G}]$. For example, we may want the identified causes to be mutually independent (which leads to a factorial representation or code) or sparse, or of lower dimension than the input data. Many heuristics can be formalized using the information theoretic ideas we discuss in chapter 4.

factorial coding
sparse coding
dimensionality reduction

Recognition Models

Once the optimal generative model has been constructed, the culmination of representational learning is recognition, in which new input data are interpreted in terms of the causes established by the generative model. In probabilistic recognition models, this amounts to determining the probability that cause v is associated with input $\mathbf{u}$, $P[v|\mathbf{u}; \mathcal{G}]$, which is called the posterior distribution over causes or the recognition distribution.

recognition distribution $P[v|\mathbf{u}; \mathcal{G}]$

In the model of figure 10.1, and in many of the models discussed in this chapter, recognition falls directly out of the generative model. The probability of cause v, given input $\mathbf{u}$, $P[v|\mathbf{u}; \mathcal{G}]$, is the statistical inverse of the

distribution $p[\mathbf{u}|v; \mathcal{G}]$ that defines the generative model. Using Bayes theorem, it can be expressed in terms of the distributions that define the generative model as

$$P[v|\mathbf{u}; \mathcal{G}] = \frac{p[\mathbf{u}|v; \mathcal{G}]P[v; \mathcal{G}]}{p[\mathbf{u}; \mathcal{G}]}. \tag{10.3}$$

Once the recognition distribution has been computed from this equation, the probability of various causes being associated with a given input can be determined. For instance, in the example of figure 10.1, equation 10.3 can be used to determine that the point indicated by the filled square in figure 10.1C has probability $P[v = \mathrm{A}|\mathbf{u}; \mathcal{G}] = 0.8$ of being associated with neuron A and $P[v = \mathrm{B}|\mathbf{u}; \mathcal{G}] = 0.2$ of being associated with neuron B.

Recall that constructing a generative model involves making a number of assumptions about the nature of the causes underlying a set of inputs. The recognition model provides a mechanism for checking the self-consistency of these assumptions. This is done by examining the distribution of causes produced by the recognition model in response to actual data. This distribution should match the prior distribution over causes, and thus share its desired properties, such as mutual independence. If the prior distribution of the generative model does not match the actual distribution of causes produced by the recognition model, this is an indication that the imposed heuristic does not apply accurately to the input data.

Expectation Maximization

EM

During our discussion of generative models, we skipped over the process by which the parameters $\mathcal{G}$ are refined to optimize the match between synthetic and real input data. There are various ways of doing this. In this chapter (except for one case), we use an approach called expectation maximization (EM). The general theory of EM is discussed in detail in the following section but, as an introduction to the method, we apply it here to the example of figure 10.1. Recall that the problem of optimizing the generative model in this case involves adjusting the mixing proportions, means, and variances of the two Gaussian distributions until the clusters of synthetic data points in figure 10.1B and 10.1C match the clusters of actual data points in figure 10.1A.

To optimize the match between synthetic and real input data, the parameters $\mathbf{g}_v$ and $\mathbf{\Sigma}_v$, for v=A and B, of the Gaussian distributions of the generative model should equal the means and variances of the data points associated with each cluster in figure 10.1A. If we knew which cluster each input point belonged to, it would be a simple matter to compute these means and variances and construct the optimal generative model. Similarly, we could set γ_v, the prior probability of a given spike being a member of cluster v, equal to the fraction of data points assigned to that cluster. Of course, we do not know the cluster assignments of the input points; that would amount to knowing the answer to the recognition problem. However, we can make an informed guess about which point belongs to which

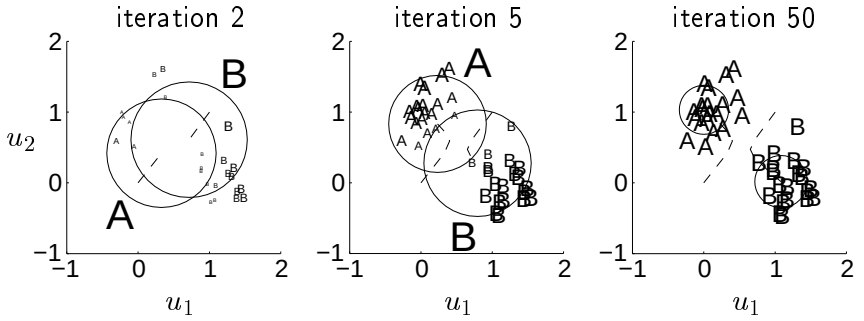

Figure 10.2 EM for clustering. Three stages during the course of EM learning of a generative model. The circles show the Gaussian distributions for clusters A and B (labeled with the largest A and B) as in figure 10.1B and 10.1C. The "trails" behind the centers of the circles plot the change in the mean since the last iteration. The data from figure 10.1A are plotted using the small labels. Label A is used if $P[v = \mathrm{A}|\mathbf{u}; \mathcal{G}] > 0.5$ (and otherwise label B), with the font size proportional to $|P[v = \mathrm{A}|\mathbf{u}; \mathcal{G}] - 0.5|$. This makes the fonts small in regions where the two distributions overlap, even inside one of the circles. The assignment of labels for the two Gaussians (i.e., which is A and which is B) depends on initial conditions.

cluster using the recognition distribution computed from equation 10.3. In other words, the recognition distribution $P[v|\mathbf{u}; \mathcal{G}]$ provides us with our best current guess about the cluster assignment, and this can be used in place of the actual knowledge about which neuron produced which spike. $P[v|\mathbf{u}; \mathcal{G}]$ is thus used to assign the data point $\mathbf{u}$ to cluster v in a probabilistic manner. In this context, $P[v|\mathbf{u}; \mathcal{G}]$ is also called the responsibility of v for $\mathbf{u}$.

responsibility

Following this reasoning, the mean and variance of the Gaussian distribution corresponding to cause v are set equal to a weighted mean and variance of all the data points, with the weight for point $\mathbf{u}$ equal to the current estimate $P[v|\mathbf{u}; \mathcal{G}]$ of the probability that it belongs to cluster v. A similar argument is applied to the mixing proportions, resulting in the equations

$$\gamma_v = \langle P[v|\mathbf{u}; \mathcal{G}] \rangle , \quad \mathbf{g}_v = \frac{\langle P[v|\mathbf{u}; \mathcal{G}]\mathbf{u} \rangle}{\gamma_v} , \quad \Sigma_v = \frac{\langle P[v|\mathbf{u}; \mathcal{G}]|\mathbf{u} - \mathbf{g}_v|^2 \rangle}{2\gamma_v} . \tag{10.4}$$

The angle brackets indicate averages over all the input data points. The factors of γ_v dividing the last two expressions correct for the fact that the number of points in cluster v is expected to be γ_v times the total number of input data points, whereas the full averages denoted by the brackets involve dividing by the total number of data points.

The full EM algorithm consists of two phases that are applied in alternation. In the E (or expectation) phase, the responsibilities $P[v|\mathbf{u}; \mathcal{G}]$ are calculated from equation 10.3. In the M (or maximization) phase, the generative parameters $\mathcal{G}$ are modified according to equation 10.4. The process of determining the responsibilities and then averaging according to them repeats because the responsibilities change when $\mathcal{G}$ is modified. Figure 10.2

E phase
M phase

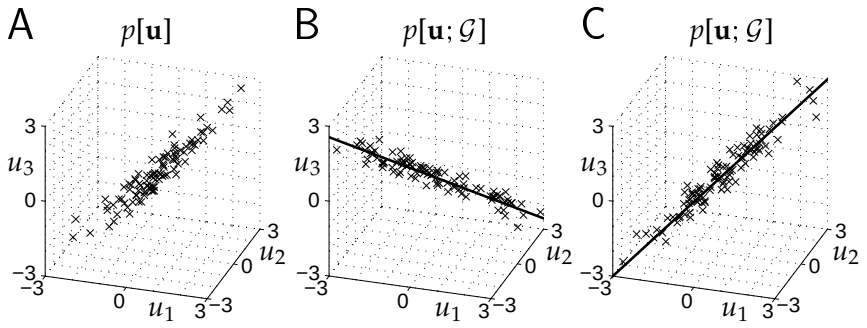

Figure 10.3 Factor analysis. (A) Input data points drawn from the distribution $p[\mathbf{u}]$ are indicated by the crosses. (B) The initial generative model. The solid line shows $\mathbf{g}$, and the crosses are synthetic data, which are samples from the generative distribution $p[\mathbf{u};\mathcal{G}]$ (with $\Sigma_a = 0.0625$, for all a). (C) The line $\mathbf{g}$ and synthetic data points generated by the optimal generative model.

shows intermediate results at three different times during the running of the EM procedure, starting from the generative model in figure 10.1B and ending up with the fit shown in figure 10.1C.

Continuous Generative Models

The data of figure 10.1A consist of two separated clusters of points, so a cause v that takes only two different values is appropriate. Figure 10.3A shows data that suggest the need for a continuous variable v. We can think of these data as the outputs of three noisy sensors, each measuring the same quantity. In this case, the cause v represents the value of the quantity being measured, and recognition corresponds to extracting this value from the sensor outputs. Because v is a continuous variable, the prior and recognition distributions in this case are probability densities, $p[v;\mathcal{G}]$ and $p[v|\mathbf{u};\mathcal{G}]$.

As for clustering, the generative model is determined by the prior distribution $p[v;\mathcal{G}]$ and the generative distribution $p[\mathbf{u}|v;\mathcal{G}]$, where $\mathbf{u} = (u_1, u_2, u_3)$ represents the three sensor readings. A simple choice for the prior distribution over v is a Gaussian with mean 0 and variance 1. The generative distribution is designed to capture the fact that the data points in figure 10.3A lie along a line in the three-dimensional space. It is the product of three Gaussian functions, one for each of the sensors, with means $g_a v$ and variances Σ_a for $a = 1, 2, 3$. The vector $\mathbf{g} = (g_1, g_2, g_3)$ specifies the direction of the line along which synthetic data points produced by the generative model lie, and the variances determine how tightly the points hug this line in each input dimension. In the example of figure 10.3A, the sensors all measure the same quantity. Thus, from an arbitrary initial $\mathbf{g}$, the generative model must find the best fit, $\mathbf{g} \propto (1, 1, 1)$.

Figure 10.3B shows synthetic data points generated from the generative

model, along with a solid line indicating the direction of $\mathbf{g}$. As in figure 10.1B, although the generative model has the capacity to create a data distribution like that in figure 10.3A, the parameters underlying figure 10.3B are clearly inappropriate, and must be adjusted by a learning procedure. Figure 10.3C shows synthetic data after learning, indicating the close match between the marginal distribution $p[\mathbf{u}; \mathcal{G}]$ from the model and the input distribution $p[\mathbf{u}]$.

This model is a simple case of factor analysis; the general case is discussed in section 10.3. The EM algorithm for factor analysis is similar in structure to that for clustering. As before, the basic idea is that if we knew the value of the cause that underlies each input point, we could find the parameters $\mathcal{G}$ easily. Here, the parameters would be determined by solving the linear regression problem that fits the observed inputs to the variable v. This mirrors the observation in our first example that if we knew the cluster assignment for each input point, we could easily find the optimal means and variances of the clusters. Of course, we do not know the values of the causes. Rather, as before, in the E phase of the EM algorithm, the distribution over causes $p[v|\mathbf{u}; \mathcal{G}]$ is calculated from the continuous analog of equation 10.3 ($P[v; \mathcal{G}]$ on the right side of equation 10.3 is replaced by $p[v; \mathcal{G}]$), and this is used as our best current estimate of how likely cause v is associated with input $\mathbf{u}$. Then the M phase consists of weighted linear regression, fitting the observations $\mathbf{u}$ to the variables v weighted by the current recognition probabilities. The result is analogous to equation 10.4; we set

$$g_a = \frac{\langle \int dv\, p[v|\mathbf{u}; \mathcal{G}] v u_a \rangle}{\langle \int dv\, p[v|\mathbf{u}; \mathcal{G}] v^2 \rangle} \quad \text{and} \quad \Sigma_a = \left\langle \int dv\, p[v|\mathbf{u}; \mathcal{G}] (u_a - v g_a)^2 \right\rangle . \tag{10.5}$$

Approximate Recognition

In the two examples we have considered, equation 10.3 was used to obtain the recognition distribution directly from the generative model. For some models, however, it is impractically difficult to evaluate the right side of equation 10.3 and obtain the recognition distribution in this way. We call models in which the recognition distribution can be computed tractably from equation 10.3 invertible, and those in which it cannot be computed tractably, noninvertible. In the latter case, because equation 10.3 cannot be used, recognition is based on an approximate recognition distribution. This is a function $Q[v; \mathbf{u}]$ that approximates the exact recognition distribution $P[v|\mathbf{u}; \mathcal{G}]$. Often, as we discuss in the next section, the best approximation of the recognition distribution comes from adjusting parameters through an optimization procedure. Once this is done, $Q[v; \mathbf{u}]$ provides the model's estimate of the probability that input $\mathbf{u}$ is associated with cause v, and substitutes for the exact recognition distribution $P[v|\mathbf{u}; \mathcal{G}]$.

invertible and noninvertible models

approximate recognition distribution $Q[v; \mathbf{u}]$

The E phase of the EM algorithm in a noninvertible model consists of

making $Q[v; \mathbf{u}]$ approximate $P[v|\mathbf{u}; \mathcal{G}]$ as accurately as possible, given the current parameters $\mathcal{G}$. We can include invertible models within the same general formalism used to describe noninvertible models by noting that, in the E phase for an invertible model, we simple set $Q[v; \mathbf{u}] = P[v|\mathbf{u}; \mathcal{G}]$ by solving equation 10.3.

Summary of Causal Models

In summary, causal models make use of the following probability distributions (for the case of continuous inputs and discrete causes).

- $p[\mathbf{u}]$, the input distribution
- $P[v; \mathcal{G}]$, the prior distribution over causes
- $p[\mathbf{u}|v; \mathcal{G}]$, the generative distribution
- $p[\mathbf{u}; \mathcal{G}]$, the marginal distribution
- $P[v|\mathbf{u}; \mathcal{G}]$, the recognition distribution
- $P[\mathbf{u}, v; \mathcal{G}]$, the joint distribution over inputs and causes
- $Q[v; \mathbf{u}]$, the approximate recognition distribution.

The goal of generative modeling, which is implemented by successive M phases of the EM algorithm, is to make $p[\mathbf{u}; \mathcal{G}] \approx p[\mathbf{u}]$ (as accurately as possible). This is done by using the marginal distribution obtained from prior E phases of EM and adjusting the parameters $\mathcal{G}$ to match it to the input distribution. The goal of each E phase is to make $Q[v; \mathbf{u}] \approx P[v|\mathbf{u}; \mathcal{G}]$ (as accurately as possible) for the current values of the generative parameters $\mathcal{G}$. Probabilistic recognition is carried out using the distribution $Q[v; \mathbf{u}]$ to determine the probability that cause v is responsible for input $\mathbf{u}$.

10.2 Density Estimation

density estimation

The process of matching the distribution $p[\mathbf{u}; \mathcal{G}]$ produced by the generative model to the actual input distribution $p[\mathbf{u}]$ is a form of density estimation. This technique is discussed in chapter 8 in connection with the Boltzmann machine. As mentioned in the introduction, the parameters $\mathcal{G}$ of the generative model are fitted to the input data by minimizing the discrepancy between the probability density of the input data $p[\mathbf{u}]$ and the marginal probability density $p[\mathbf{u}; \mathcal{G}]$ of equation 10.1. This discrepancy is measured using the Kullback-Leibler divergence (chapter 4),

$$\begin{aligned} D_{\mathrm{KL}}(p[\mathbf{u}], p[\mathbf{u}; \mathcal{G}]) &= \int d\mathbf{u}\, p[\mathbf{u}] \ln \frac{p[\mathbf{u}]}{p[\mathbf{u}; \mathcal{G}]} \\ &\approx -\langle \ln p[\mathbf{u}; \mathcal{G}] \rangle + K\,, \end{aligned} \tag{10.6}$$

where K is a term associated with the entropy of the distribution $p[\mathbf{u}]$, that is independent of $\mathcal{G}$. In the second line, we have approximated the integral over all $\mathbf{u}$ values weighted by $p[\mathbf{u}]$ by the average over input data points generated from the distribution $p[\mathbf{u}]$. We assume there are sufficient input data to justify this approximation.

As in the case of the Boltzmann machine discussed in chapter 8, equation 10.6 implies that minimizing the discrepancy between $p[\mathbf{u}]$ and $p[\mathbf{u}; \mathcal{G}]$ amounts to maximizing the log likelihood that the training data could have been created by the generative model,

log likelihood $L(\mathcal{G})$

$$L(\mathcal{G}) = \langle \ln p[\mathbf{u}; \mathcal{G}] \rangle . \tag{10.7}$$

$L(\mathcal{G})$ is the average log likelihood, and the method is known as maximum likelihood density estimation. A theorem due to Shannon describes circumstances under which the generative model that maximizes the likelihood over input data also provides the most efficient way of coding those data, so density estimation is closely related to optimal coding.

maximum likelihood density estimation

Theory of EM

Although stochastic gradient ascent can be used to adjust the parameters of the generative model to maximize the likelihood in equation 10.7 (as it was for the Boltzmann machine), the EM algorithm discussed in the introduction is an alternative procedure that is often more efficient. We applied this algorithm, on intuitive grounds, to the examples of figures 10.1 and 10.3, but we now present a more general and rigorous discussion. This is based on the connection of EM with maximization of the function

$\mathcal{F}(Q, \mathcal{G})$

$$\mathcal{F}(Q, \mathcal{G}) = \left\langle \sum_{v} Q[v; \mathbf{u}] \ln \frac{p[v, \mathbf{u}; \mathcal{G}]}{Q[v; \mathbf{u}]} \right\rangle , \tag{10.8}$$

where $Q[v; \mathbf{u}]$ is any nonnegative function of the discrete argument v and continuous input $\mathbf{u}$ that satisfies

$$\sum_{v} Q[v; \mathbf{u}] = 1 \tag{10.9}$$

for all $\mathbf{u}$. Although, in principle, $Q[v; \mathbf{u}]$ can be any function, we consider it to be an approximate recognition distribution, that is $Q[v; \mathbf{u}] \approx P[v|\mathbf{u}; \mathcal{G}]$.

$\mathcal{F}$ is a useful quantity because, by a rearrangement of terms, it can be written as the difference of the average log likelihood and the average Kullback-Leibler divergence between $Q[v; \mathbf{u}]$ and $P[v|\mathbf{u}; \mathcal{G}]$. This is done by noting that the joint distribution over inputs and causes satisfies $p[v, \mathbf{u}; \mathcal{G}] = P[v|\mathbf{u}; \mathcal{G}]p[\mathbf{u}; \mathcal{G}]$, in addition to 10.2, and using 10.9 and the

definition of the Kullback-Leibler divergence to obtain

$$\begin{aligned}\mathcal{F}(Q, \mathcal{G}) &= \left\langle \sum_v Q[v; \mathbf{u}] \left(\ln p[\mathbf{u}; \mathcal{G}] + \ln \frac{P[v|\mathbf{u}; \mathcal{G}]}{Q[v; \mathbf{u}]} \right) \right\rangle \\ &= \langle \ln p[\mathbf{u}; \mathcal{G}] \rangle - \left\langle \sum_v Q[v; \mathbf{u}] \left(\ln \frac{Q[v; \mathbf{u}]}{P[v|\mathbf{u}; \mathcal{G}]} \right) \right\rangle \\ &= L(\mathcal{G}) - \left\langle D_{\mathrm{KL}}(Q[v; \mathbf{u}], P[v|\mathbf{u}; \mathcal{G}]) \right\rangle . \end{aligned} \tag{10.10}$$

Because the Kullback-Leibler divergence is never negative,

$$L(\mathcal{G}) \geq \mathcal{F}(Q, \mathcal{G}) , \tag{10.11}$$

and because $D_{\mathrm{KL}} = 0$ only if the two distributions being compared are identical, this inequality is saturated, becoming an equality, only if

$$Q[v; \mathbf{u}] = P[v|\mathbf{u}; \mathcal{G}] . \tag{10.12}$$

free energy $-\mathcal{F}$

The negative of $\mathcal{F}$ is related to the free energy used in statistical physics.

Expressions 10.10, 10.11, and 10.12 are critical to the operation of EM. The two phases of EM are concerned with separately maximizing (or at least increasing) $\mathcal{F}$ with respect to one of its two arguments, keeping the other one fixed. When $\mathcal{F}$ increases, this increases a lower bound on the log likelihood of the input data (equation 10.11). In the M phase, $\mathcal{F}$ is increased with respect to $\mathcal{G}$, keeping Q constant. For the generative models considered as examples in the previous section, it is possible to maximize $\mathcal{F}$ with respect to $\mathcal{G}$ in a single step. For other generative models, this may require multiple steps that perform gradient ascent on $\mathcal{F}$. In the E phase, $\mathcal{F}$ is increased with respect to Q, keeping $\mathcal{G}$ constant. From equation 10.10, we see that increasing $\mathcal{F}$ by changing Q is equivalent to reducing the average Kullback-Leibler divergence between $Q[v; \mathbf{u}]$ and $P[v|\mathbf{u}; \mathcal{G}]$. This makes $Q[v; \mathbf{u}]$ a better approximation of $P[v|\mathbf{u}; \mathcal{G}]$. The E phase can proceed in at least three possible ways, depending on the nature of the generative model being considered. We discuss these separately.

One advantage of EM over likelihood maximization through gradient methods is that large steps toward the maximum can be taken during each M cycle of modification. Of course, the log likelihood may have multiple maxima, in which case neither gradient ascent nor EM is guaranteed to find the globally optimal solution.

Invertible Models

If the causal model being considered is invertible, the E step of EM simply consists of solving equation 10.3 for the recognition distribution, and setting Q equal to the resulting $P[v|\mathbf{u}; \mathcal{G}]$, as in equation 10.12. This maximizes $\mathcal{F}$ with respect to Q by setting the Kullback-Leibler term in equation 10.10 to 0, and it makes the function $\mathcal{F}$ equal to $L(\mathcal{G})$, the average log

likelihood of the data points. However, the EM algorithm for maximizing $\mathcal{F}$ is not exactly the same as likelihood maximization by gradient ascent of $\mathcal{F}$. This is because the function Q is held constant during the M phase while the parameters of the generative model are modified. Although $\mathcal{F}$ is equal to L at the beginning of the M phase, exact equality ceases to be true as soon as the parameters are modified, making $P[v|\mathbf{u}; \mathcal{G}]$ different from Q. $\mathcal{F}$ is equal to $L(\mathcal{G})$ again only after the update of Q during the following E phase. At this point, $L(\mathcal{G})$ must have increased since the last E phase, because $\mathcal{F}$ has increased. This shows that the log likelihood increases monotonically during EM until the process converges.

For the example of figure 10.1, the joint probability over causes and inputs is

$$p[v, \mathbf{u}; \mathcal{G}] = \frac{\gamma_v}{2\pi\Sigma_v} \exp\left(-\frac{|\mathbf{u} - \mathbf{g}_v|^2}{2\Sigma_v}\right), \tag{10.13}$$

and thus

$$\mathcal{F} = \left\langle \sum_v Q[v; \mathbf{u}] \left(\ln\left(\frac{\gamma_v}{2\pi}\right) - \ln \Sigma_v - \frac{|\mathbf{u} - \mathbf{g}_v|^2}{2\Sigma_v} - \ln Q[v; \mathbf{u}] \right) \right\rangle . \tag{10.14}$$

The E phase amounts to computing $P[v|\mathbf{u}; \mathcal{G}]$ from equation 10.3 and setting Q equal to it, as in equation 10.12. The M phase involves maximizing $\mathcal{F}$ with respect to $\mathcal{G}$ for this Q. We leave it as an exercise for the reader to show that maximizing equation 10.14 with respect to the parameters γ_v (taking into account the constraint $\sum_v \gamma_v = 1$), $\mathbf{g}_v$, and Σ_v leads to the rules of equation 10.4. For the example of figure 10.3, the joint probability is

$$p[v, \mathbf{u}; \mathcal{G}] = \frac{\exp(-v^2/2)}{\sqrt{2\pi}} \frac{\exp(-\sum_a (u_a - g_a v)^2 / 2\Sigma_a)}{\sqrt{(2\pi)^3 \Sigma_1 \Sigma_2 \Sigma_3}}, \tag{10.15}$$

from which it is straightforward to calculate the relevant $\mathcal{F}$ function and the associated learning rules of equation 10.5.

Noninvertible Deterministic Models

If the generative model is noninvertible, the E phase of the EM algorithm is more complex than simply setting Q equal to $P[v|\mathbf{u}; \mathcal{G}]$, because it is not practical to compute the recognition distribution exactly. The steps taken during the E phase depend on whether the approximation to the inverse of the model is deterministic or probabilistic, although the basic argument is the same in either case.

Deterministic recognition results in a prediction $v(\mathbf{u})$ of the cause underlying input $\mathbf{u}$. In terms of the function $\mathcal{F}$, this amounts to retaining only the single term $v = v(\mathbf{u})$ in the sum in equation 10.8, and for this single term $Q[v(\mathbf{u}); \mathbf{u}] = 1$. Thus, in this case $\mathcal{F}$ is a functional of the function $v(\mathbf{u})$ and a function of the parameters $\mathcal{G}$ given by

$$\mathcal{F}(Q, \mathcal{G}) = \mathcal{F}(v(\mathbf{u}), \mathcal{G}) = \langle \ln P[v(\mathbf{u}), \mathbf{u}; \mathcal{G}] \rangle . \tag{10.16}$$

The M phase of EM consists, as always, of maximizing this expression with respect to $\mathcal{G}$. During the E phase we try to find the function $v(\mathbf{u})$ that maximizes $\mathcal{F}$. Because v is varied during the optimization procedure, the approach is sometimes called a variational method. The E and M steps make intuitive sense; we are finding the input-output relationship that maximizes the probability that the generative model would have simultaneously produced the input $\mathbf{u}$ and cause $v(\mathbf{u})$.

variational method

Noninvertible Probabilistic Models

The alternative to using a deterministic approximate recognition model is to treat $Q[v; \mathbf{u}]$ as a full probability distribution over v for each input example $\mathbf{u}$. In this case, we choose a specific functional form for Q, expressed in terms of a set of parameters collectively labeled $\mathcal{W}$. Thus, we write the approximate recognition distribution as $Q[v; \mathbf{u}, \mathcal{W}]$. Like generative models, approximate recognition models can have different structures and parameters. $\mathcal{F}$ can now be treated as a function of $\mathcal{W}$, rather than of Q, so we write it as $\mathcal{F}(\mathcal{W}, \mathcal{G})$. As in all cases, the M phase of EM consists of maximizing $F(\mathcal{W}, \mathcal{G})$ with respect to $\mathcal{G}$. The E phase now consists of maximizing $F(\mathcal{W}, \mathcal{G})$ with respect to $\mathcal{W}$. This has the effect of making $Q[v; \mathbf{u}, \mathcal{W}]$ as similar as possible to $P[v|\mathbf{u}; \mathcal{G}]$, in the sense that the KL divergence between them, averaged over the input data, is minimized (see equation 10.10).

In some cases, $\mathcal{W}$ has separate parameters for each possible input $\mathbf{u}$. This means that each input has a separate approximate recognition distribution which is individually tailored, subject to the inherent simplifying assumptions, to its own causes. The mean-field approximation to the Boltzmann machine discussed in chapter 7 is an example of this type.

It is not necessary to maximize $F(\mathcal{W}, \mathcal{G})$ completely with respect to $\mathcal{W}$ and then $\mathcal{G}$ during successive E and M phases. Instead, gradient ascent steps that modify $\mathcal{W}$ and $\mathcal{G}$ by small amounts can be taken in alternation, in which case the E and M phases effectively overlap.

Because each E and M step separately increases the value of $\mathcal{F}$, the EM algorithm is guaranteed to converge to at least a local maximum of $\mathcal{F}$, except in rare cases when the process of maximizing a function one coordinate at a time (which is called coordinate ascent) finds local maxima that other optimization methods avoid (we encounter an example of this later in the chapter). In general, the maximum found does not correspond to a local maximum of the likelihood function because Q is not exactly equal to the actual recognition distribution (that is, $\mathcal{F}$ is guaranteed only to be a lower bound on $L(\mathcal{G})$). Nevertheless, a good generative model should be obtained if the lower bound is tight.

10.3 Causal Models for Density Estimation

In this section, we present a number of models in which representational learning is achieved through density estimation. The mixture of Gaussians and factor analysis models that we have already mentioned are examples of invertible generative models with probabilistic recognition. *K*-means is a limiting case of mixture of Gaussians with deterministic recognition, and principal components analysis is a limiting case of factor analysis with deterministic recognition. We consider two other models with deterministic recognition: independent components analysis, which is invertible; and sparse coding, which is noninvertible. Our final example, the Helmholtz machine, is noninvertible with probabilistic recognition. The Boltzmann machine, discussed in chapters 7 and 8, is an additional example that is closely related to the causal models discussed here. We summarize and interpret general properties of representations derived from causal models at the end of the chapter. The table in the appendix summarizes the generative and recognition distributions and the learning rules for all the models we discuss.

Mixture of Gaussians

The model applied in the introduction to the data in figure 10.1A is a mixture of Gaussians model. That example involved two causes and two Gaussian distributions, but we now generalize this to N_v causes, each associated with a separate Gaussian distribution. The model is defined by the probability distributions

$$P[v; \mathcal{G}] = \gamma_v \quad \text{and} \quad p[\mathbf{u}|v; \mathcal{G}] = \mathcal{N}(\mathbf{u}; \mathbf{g}_v, \Sigma_v), \tag{10.17}$$

where v takes N_v values representing the different causes and, for an N_u component input vector,

$$\mathcal{N}(\mathbf{u}; \mathbf{g}, \Sigma) = \frac{1}{(2\pi\Sigma)^{N_u/2}} \exp\left(-\frac{|\mathbf{u}-\mathbf{g}|^2}{2\Sigma}\right) \tag{10.18}$$

is a Gaussian distribution with mean $\mathbf{g}$ and variances for the individual components equal to Σ. The function $\mathcal{F}(Q, \mathcal{G})$ for this model is given by an expression similar to equation 10.14 (with slightly different factors if $N_u \neq 2$), leading to the M-phase learning rules given in the appendix. Once the generative model has been optimized, the recognition distribution is constructed from equation 10.3 as

$$P[v|\mathbf{u}; \mathcal{G}] = \frac{\gamma_v \mathcal{N}(\mathbf{u}; \mathbf{g}_v, \Sigma_v)}{\sum_{v'} \gamma_{v'} \mathcal{N}(\mathbf{u}; \mathbf{g}_{v'}, \Sigma_{v'})}. \tag{10.19}$$

K-Means Algorithm

A special case of mixture of Gaussians can be derived in the limit that the variances of the Gaussians are equal and tend toward 0, $\Sigma_v = \Sigma \to 0$. We

discuss this limit for two clusters, as in figure 10.1. When Σ is extremely small, the recognition distribution $P[v|\mathbf{u}; \mathcal{G}]$ of equation 10.19 degenerates because it takes essentially two values, 0 or 1, depending on whether $\mathbf{u}$ is closer to one cluster or the other. This provides a deterministic, rather than a probabilistic, classification of $\mathbf{u}$. In the degenerate case, EM consists of choosing two random values for the centers of the two cluster distributions, and then repeatedly finding all the inputs $\mathbf{u}$ that are closest to a given center $\mathbf{g}_v$, and then moving $\mathbf{g}_v$ to the average of these points. This is called the K-means algorithm (with $K = 2$ for two clusters). The mixing proportions γ_v do not play an important role for the K-means algorithm. New input points are recognized as belonging to the clusters to which they are closest.

Factor Analysis

The model used in figure 10.3 is an example of factor analysis. In general, factor analysis involves a continuous vector of causes, $\mathbf{v}$, drawn from a Gaussian prior distribution, and uses a Gaussian generative distribution with a mean that depends linearly on $\mathbf{v}$. We assume that the distribution $p[\mathbf{u}]$ has a mean of 0 (nonzero means can be accommodated simply by shifting the input data). The defining distributions for factor analysis are thus

$$p[\mathbf{v}; \mathcal{G}] = \mathcal{N}(\mathbf{v}; \mathbf{0}, 1) \quad \text{and} \quad p[\mathbf{u}|\mathbf{v}; \mathcal{G}] = \mathcal{N}(\mathbf{u}; \mathbf{G} \cdot \mathbf{v}, \mathbf{\Sigma}), \tag{10.20}$$

where the extension of equation 10.18, expressed in terms of the mean $\mathbf{g}$ and covariance matrix $\mathbf{\Sigma}$, is

$$\mathcal{N}(\mathbf{u}; \mathbf{g}, \mathbf{\Sigma}) = \frac{1}{((2\pi)^{N_u} |\det \mathbf{\Sigma}|)^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{u} - \mathbf{g}) \cdot \mathbf{\Sigma}^{-1} \cdot (\mathbf{u} - \mathbf{g})\right). \tag{10.21}$$

The expression $|\det \mathbf{\Sigma}|$ indicates the (absolute) value of the determinant of $\mathbf{\Sigma}$. In factor analysis, $\mathbf{\Sigma}$ is taken to be diagonal, $\mathbf{\Sigma} = \mathrm{diag}(\Sigma_1, \ldots, \Sigma_{N_u})$ (see the Mathematical Appendix), with all the diagonal elements strictly positive, so its inverse is simply $\mathbf{\Sigma}^{-1} = \mathrm{diag}(1/\Sigma_1, \ldots, 1/\Sigma_{N_u})$ and $|\det \mathbf{\Sigma}| = \Sigma_1 \Sigma_2 \ldots \Sigma_{N_u}$.

Because $\mathbf{\Sigma}$ is diagonal, the individual components of $\mathbf{v}$ are mutually independent. Thus, any correlations between the components of $\mathbf{u}$ must arise from the mean values $\mathbf{G} \cdot \mathbf{v}$ of the generative distribution. To be well specified, the model requires $\mathbf{v}$ to have fewer dimensions than $\mathbf{u}$ ($N_v < N_u$). In terms of heuristics, factor analysis seeks a relatively small number of independent causes that account, in a linear manner, for collective Gaussian structure in the inputs.

The recognition distribution for factor analysis has the Gaussian form

$$p[\mathbf{v}|\mathbf{u}; \mathcal{G}] = \mathcal{N}(\mathbf{v}; \mathbf{W} \cdot \mathbf{u}, \mathbf{\Psi}), \tag{10.22}$$

where expressions for $\mathbf{W}$ and $\mathbf{\Psi}$ are given in the appendix. These do not depend on the input $\mathbf{u}$, so factor analysis involves a linear relation between the input and the mean of the recognition distribution. EM, as applied to an invertible model, can be used to adjust $\mathcal{G} = (\mathbf{G}, \mathbf{\Sigma})$ on the basis of the input data. The resulting learning rules are given in the appendix. For the case of a single cause v, these reduce to equation 10.5.

In this case, we can understand the goal of density estimation in an additional way. By direct calculation, as in equation 10.1, the marginal distribution for $\mathbf{u}$ is

$$p[\mathbf{u}; \mathcal{G}] = \mathcal{N}(\mathbf{u}; \mathbf{0}, \mathbf{G} \cdot \mathbf{G}^{\mathrm{T}} + \mathbf{\Sigma}), \tag{10.23}$$

where $[\mathbf{G}^{\mathrm{T}}]_{ab} = [\mathbf{G}]_{ba}$ and $[\mathbf{G} \cdot \mathbf{G}^{\mathrm{T}}]_{ab} = \sum_c G_{ac} G_{bc}$ (see the Mathematical Appendix). Maximum likelihood density estimation requires determining the $\mathcal{G}$ that makes $\mathbf{G} \cdot \mathbf{G}^{\mathrm{T}} + \mathbf{\Sigma}$ match, as closely as possible, the covariance matrix of the input distribution.

Principal Components Analysis

In the same way that setting the parameters Σ_v to 0 in the mixture of Gaussians model leads to the K-means algorithm, setting all the variances in factor analysis to 0 leads to another well-known method, principal components analysis (which we also discuss in chapter 8). To see this, consider the case of a single factor. This means that v is a single number, and that the mean of the distribution $p[\mathbf{u}|v; \mathcal{G}]$ is $v\mathbf{g}$, where the vector $\mathbf{g}$ replaces the matrix $\mathbf{G}$ of the general case. The elements of the diagonal matrix $\mathbf{\Sigma}$ are set to a single variance Σ, which we shrink to 0.

As $\Sigma \to 0$, the Gaussian distribution $p[\mathbf{u}|v; \mathcal{G}]$ in equation 10.20 approaches a δ function (see the Mathematical Appendix), and it can generate only the single vector $\mathbf{u}(v) = v\mathbf{g}$ from cause v. Similarly, the recognition distribution of equation 10.22 becomes a δ function, making the recognition process deterministic with $v(\mathbf{u}) = \mathbf{W} \cdot \mathbf{u}$ given by the mean of the recognition distribution of equation 10.22. Using the expression for $\mathbf{W}$ in the appendix in the limit $\Sigma \to 0$, we find

$$v(\mathbf{u}) = \frac{\mathbf{g} \cdot \mathbf{u}}{|\mathbf{g}|^2}. \tag{10.24}$$

This is the result of the E phase of EM. In the M phase, we maximize

$$\mathcal{F}(v(\mathbf{u}), \mathcal{G}) = \langle \ln p[v(\mathbf{u}), \mathbf{u}; \mathcal{G}] \rangle = K - \frac{N_u \ln \Sigma}{2} - \left\langle \frac{v^2(\mathbf{u})}{2} + \frac{|\mathbf{u} - \mathbf{g}v(\mathbf{u})|^2}{2\Sigma} \right\rangle \tag{10.25}$$

with respect to $\mathbf{g}$, without changing the expression for $v(\mathbf{u})$. Here, K is a term independent of $\mathbf{g}$ and Σ. In this expression, the only term that depends on $\mathbf{g}$ is proportional to $|\mathbf{u} - \mathbf{g}v(\mathbf{u})|^2$. Minimizing this in the M

phase produces a new value of $\mathbf{g}$ given by

$$\mathbf{g} = \frac{\langle v(\mathbf{u})\mathbf{u}\rangle}{\langle v^2(\mathbf{u})\rangle}\,. \tag{10.26}$$

This depends only on the covariance matrix of the input distribution, as does the more general form given in the appendix. Under EM, equations 10.24 and 10.26 are alternated until convergence.

For principal components analysis, we can say more about the value of $\mathbf{g}$ at convergence. We consider the case $|\mathbf{g}|^2 = 1$ because we can always multiply $\mathbf{g}$ and divide $v(\mathbf{u})$ by the same factor to make this true without affecting the dominant term in $\mathcal{F}(v(\mathbf{u}), \mathcal{G})$ as $\Sigma \to 0$. Then, the $\mathbf{g}$ that maximizes this dominant term must minimize

$$\left\langle |\mathbf{u} - \mathbf{g}(\mathbf{g}\cdot\mathbf{u})|^2 \right\rangle = \left\langle |\mathbf{u}|^2 - (\mathbf{g}\cdot\mathbf{u})^2 \right\rangle\,. \tag{10.27}$$

Here, we have used expression 10.24 for $v(\mathbf{u})$. Minimizing 10.27 with respect to $\mathbf{g}$, subject to the constraint $|\mathbf{g}|^2 = 1$, gives the result that $\mathbf{g}$ is the eigenvector of the covariance matrix $\langle \mathbf{u}\mathbf{u}\rangle$ with maximum eigenvalue. This is just the principal component vector and is equivalent to finding the vector of unit length with the largest possible average projection onto $\mathbf{u}$. Note that there are ways other than EM for finding eigenvectors of this matrix.

The argument we have given shows that principal components analysis is a degenerate form of factor analysis. This is also true if more than one factor is considered, although maximizing $\mathcal{F}$ constrains the projections $\mathbf{G}\cdot\mathbf{u}$ and therefore is sufficient only to force $\mathbf{G}$ to represent the principal components subspace of the data. The same subspace emerges from full factor analysis provided that the variances of all the factors are equal, even when they are nonzero.

Figure 10.4 illustrates an important difference between factor analysis and principal components analysis. In this example, $\mathbf{u}$ is a three-component input vector, $\mathbf{u} = (u_1, u_2, u_3)$. Just as in figure 10.3, samples of input data were generated on the basis of a "true" cause, v_{true} according to

$$u_b = v_{\text{true}} + \epsilon_b\,, \tag{10.28}$$

where ϵ_b represents the noise added to component b of the input. Input data points were generated from this equation by choosing a value of v_{true} from a Gaussian distribution with mean 0 and variance 1, and values of ϵ_b from independent Gaussian distributions with 0 means. The variances of the distributions for ϵ_1, ϵ_2, and ϵ_3 were all are equal to 0.25 in figures 10.4A and 10.4B. However, in figures 10.4C and 10.4D, the variance for ϵ_3 is much larger (equal to 9), as if the third sensor was much noisier than the other two sensors. The graphs in figure 10.4 show the mean of the values of v extracted from sample inputs by factor analysis, or the value of v for principal components analysis, as a function of the true value used to generate the data. Perfect extraction of the underlying cause would have $v = v_{\text{true}}$, but this is impossible in this case because of the noise. The

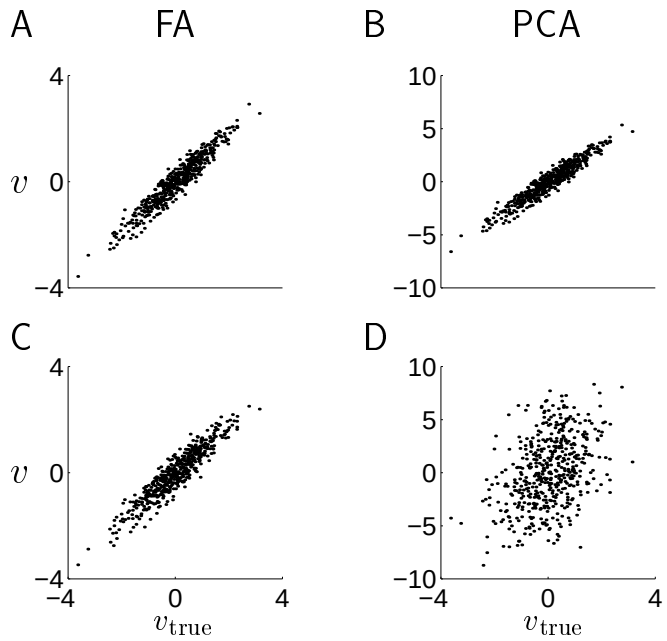

Figure 10.4 Factor analysis (FA) and principal components analysis (PCA) applied to 500 samples of noisy input reflecting a single underlying cause v_{true}. For A and B, $\langle u_i u_j \rangle = 1 + 0.25\delta_{ij}$, whereas for C and D, one sensor is corrupted by independent noise with standard deviation 3 rather than 0.5. The plots compare the values of the true cause v_{true} and the cause v inferred by the model.

best we can expect is for the v values to be well correlated with the values of v_{true}. When the input components are equally variable (figure 10.4A and 10.4B), this is indeed what happens for both factor and principal components analysis. However, when u_3 is much more variable than the other components, principal components analysis (figure 10.4D) is fooled by the extra variance and finds a cause v that does not correlate very well with v_{true}. On the other hand, factor analysis (figure 10.4C) is affected only by the covariance between the input components and not by their individual variances (which are absorbed into $\mathbf{\Sigma}$), so the cause it finds is not significantly perturbed (merely slightly degraded) by the added sensor noise.

In chapter 8, we noted that principal components analysis maximizes the mutual information between the input and output under the assumption of a linear Gaussian model. This property, and the fact that principal components analysis minimizes the reconstruction error of equation 10.27, have themselves been suggested as goals for representational learning. We have now shown how they are also related to density estimation.

Both principal components analysis and factor analysis produce a marginal distribution $p[\mathbf{u}; \mathcal{G}]$ that is Gaussian. If the actual input distribution $p[\mathbf{u}]$ is non-Gaussian, the best that these models can do is to match the mean and covariance of $p[\mathbf{u}]$; they will fail to match higher-order moments. If the input is whitened to increase coding efficiency, as discussed

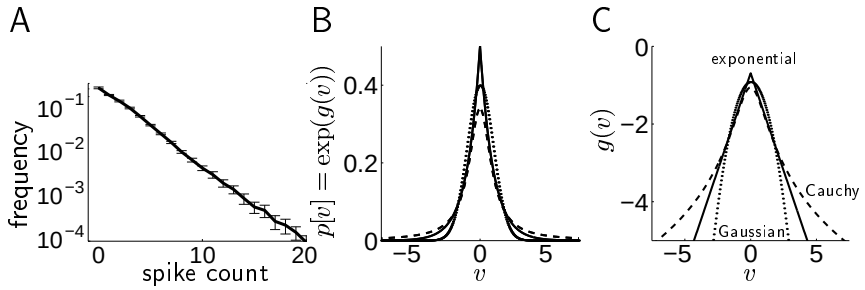

Figure 10.5 Sparse distributions. (A) Log frequency distribution for the activity of a macaque IT cell in response to video images. The fraction of times that various numbers of spikes appeared in a spike-counting window is plotted against the number of spikes. The size of the window was adjusted so that, on average, there were two spikes per window. (B) Three distributions $p[v] = \exp(g(v))$: double exponential ($g(v) = -|v|$, solid, kurtosis = 3); Cauchy ($g(v) = -\ln(1 + v^2)$, dashed, kurtosis = ∞); and Gaussian ($g(v) = -v^2/2$, dotted, kurtosis = 0). (C) The logarithms of the same three distributions. (A adapted from Baddeley et al., 1997.)

in chapter 4, so that the covariance matrix $\langle \mathbf{uu} \rangle$ is equal to the identity matrix, neither method will extract any structure at all from the input data. By contrast, the generative models discussed in the following sections produce non-Gaussian marginal distributions and attempt to account for structure in the input data beyond merely covariance (and the mean).

Sparse Coding

The v values in response to input in factor and principal components analysis tend to be Gaussian distributed. If we attempt to relate such causal variables to the activities of cortical neurons, we find a discrepancy, because the activity distributions of cortical cells in response to natural inputs are not Gaussian. Figure 10.5A shows an example of the distribution of the numbers of spikes counted within a particular time window for a neuron in the inferotemporal (IT) area of the macaque brain recorded while a monkey freely viewed television shows. The distribution is close to being exponential. This means that the neurons are most likely to fire a small number of spikes in the counting interval, but that they can occasionally fire a large number of spikes.

sparse distributions

Distributions that generate values for the components of $\mathbf{v}$ close to 0 most of the time, but occasionally far from 0, are called sparse. Intuitively, sparse distributions are more likely than Gaussians of the same mean and variance to generate values near 0, and also more likely to generate values far from 0. These occasional high values can convey substantial information. Distributions with this character are also called heavy-tailed. Figures 10.5B and 10.5C compare two sparse distributions with a Gaussian distribution.

More formally, sparseness has been defined in a variety of ways. Sparseness of a distribution is sometimes linked to a high value of a measure called kurtosis. Kurtosis of a distribution $p[v]$ is defined as *kurtosis*

$$k = \frac{\int dv\, p[v](v - \overline{v})^4}{\left(\int dv\, p[v](v - \overline{v})^2\right)^2} - 3 \quad \text{with} \quad \overline{v} = \int dv\, p[v]v\,, \tag{10.29}$$

and it takes the value 0 for a Gaussian distribution. Positive values of k are taken to imply sparse distributions, which are also called super-Gaussian or leptokurtotic. Distributions with $k < 0$ are called sub-Gaussian or platykurtotic. This is a slightly different definition of sparseness from being heavy-tailed.

A sparse representation over a large population of neurons might more naturally be defined as one in which each input is encoded by a small number of the neurons in the population. Unfortunately, identifying this form of sparseness experimentally is difficult.

Sparse coding can arise in generative models that have sparse prior distributions over causes. Unlike factor analysis and principal components analysis, sparse coding does not stress minimizing the number of representing units (i.e., components of $\mathbf{v}$), and sparse representations may require large numbers of units. This is not a disadvantage for modeling the visual system because representations in visual areas are indeed greatly expanded at various steps along the pathway. For example, there are around 40 cells in primary visual cortex for each cell in the visual thalamus. Downstream processing can benefit greatly from sparse representations because, for one thing, they minimize interference between different patterns of input.

Because they employ Gaussian priors, factor analysis and principal components analysis do not generate sparse representations. The mixture of Gaussians model is extremely sparse because each input is represented by a single cause. This may be reasonable for relatively simple input patterns, but for complex stimuli such as images, we seek something between these extremes. Olshausen and Field (1996, 1997) suggested such a model by considering a nonlinear version of factor analysis. In this model, the distribution of $\mathbf{u}$ given $\mathbf{v}$ is Gaussian with a diagonal covariance matrix, as for factor analysis, but the prior distribution over causes is sparse. Defined in terms of a function $g(v)$ (as in figure 10.5), the model has

$$p[\mathbf{v}; \mathcal{G}] \propto \prod_{a=1}^{N_v} \exp(g(v_a)) \quad \text{and} \quad p[\mathbf{u}|\mathbf{v}; \mathcal{G}] = \mathcal{N}(\mathbf{u}; \mathbf{G} \cdot \mathbf{v}, \mathbf{\Sigma})\,. \tag{10.30}$$

The prior $p[\mathbf{v}; \mathcal{G}]$ should be normalized so that its integral over $\mathbf{v}$ is 1, but we omit the normalization factor to simplify the equations.

The prior $p[\mathbf{v}; \mathcal{G}]$ in equation 10.30 makes the components of $\mathbf{v}$ mutually independent because it is a product. If we took $g(v) = -v^2$, $p[\mathbf{v}; \mathcal{G}]$ would be Gaussian (dotted lines in figures 10.5B and 10.5C), and the model would

perform factor analysis. An example of a function that provides a sparse prior is $g(v) = -\alpha|v|$. This generates a double exponential distribution (solid lines in figures 10.5B and 10.5C) similar to the activity distribution in figure 10.5A. Another commonly used form is

double exponential distribution

$$g(v) = -\ln(\beta^2 + v^2) \tag{10.31}$$

with β a constant, which generates a Cauchy distribution (dashed lines in figures 10.5B and 10.5C).

Cauchy distribution

For $g(v)$ such as equation 10.31, it is difficult to compute the recognition distribution $p[\mathbf{v}|\mathbf{u}; \mathcal{G}]$ exactly. This makes the sparse model noninvertible. Olshausen and Field chose a deterministic approximate recognition model. Thus, EM consists of finding $\mathbf{v}(\mathbf{u})$ during the E phase, and using it to adjust the parameters $\mathcal{G}$ during the M phase. To simplify the discussion, we make the covariance matrix of the generative model proportional to the identity matrix, $\mathbf{\Sigma} = \Sigma\mathbf{I}$. The function to be maximized is then

$$\mathcal{F}(\mathbf{v}(\mathbf{u}), \mathcal{G}) = \left\langle -\frac{1}{2\Sigma}|\mathbf{u} - \mathbf{G}\cdot\mathbf{v}(\mathbf{u})|^2 + \sum_{a=1}^{N_v} g(v_a(\mathbf{u})) \right\rangle + K, \tag{10.32}$$

where K is a term that is independent of $\mathbf{G}$ and $\mathbf{v}$. For convenience in discussing the EM procedure, we further take $\Sigma = 1$ and do not allow it to vary. Similarly, we assume that β in equation 10.31 is predetermined and held fixed. Then, $\mathcal{G}$ consists only of the matrix $\mathbf{G}$.

The E phase of EM involves maximizing $\mathcal{F}$ with respect to $\mathbf{v}(\mathbf{u})$ for every $\mathbf{u}$. This leads to the conditions (for all a)

$$\sum_{b=1}^{N_u}[\mathbf{u} - \mathbf{G}\cdot\mathbf{v}(\mathbf{u})]_b G_{ba} + g'(v_a) = 0\,. \tag{10.33}$$

The prime on $g(v_a)$ indicates a derivative. One way to solve this equation is to let $\mathbf{v}$ evolve over time according to the equation

$$\tau_{\mathbf{v}}\frac{dv_a}{dt} = \sum_{b=1}^{N_u}[\mathbf{u} - \mathbf{G}\cdot\mathbf{v}(\mathbf{u})]_b G_{ba} + g'(v_a)\,, \tag{10.34}$$

where $\tau_{\mathbf{v}}$ is an appropriate time constant. This equation changes $\mathbf{v}$ so that it asymptotically approaches a value $\mathbf{v} = \mathbf{v}(\mathbf{u})$ that satisfies equation 10.33 and sets the right side of equation 10.34 to 0. We assume that the evolution of $\mathbf{v}$ according to equation 10.34 is carried out long enough during the E phase for this to happen. This process is guaranteed to find only a local, not a global, maximum of $\mathcal{F}$, and it is not guaranteed to find the same local maximum on each iteration.

Equation 10.34 resembles the equation used in chapter 7 for a firing-rate network model. The term $\sum_b u_b G_{ba}$, which can be written in vector form as $\mathbf{G}^{\mathrm{T}}\cdot\mathbf{u}$, acts as the total input arising from units with activities $\mathbf{u}$ fed through a feedforward coupling matrix $\mathbf{G}^{\mathrm{T}}$. The term $-\sum_b[\mathbf{G}\cdot\mathbf{v}]_b G_{ba}$ can

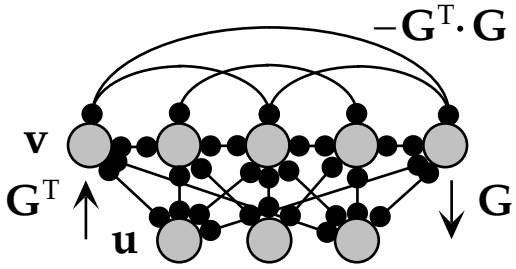

Figure 10.6 A network for sparse coding. This network reproduces equation (10.34), using recurrent weights $-\mathbf{G}^{\mathrm{T}} \cdot \mathbf{G}$ in the $\mathbf{v}$ layer and weights connecting the input units to this layer that are given by the transpose of the matrix $\mathbf{G}$. The reverse connections from the $\mathbf{v}$ layer to the input layer indicate how the mean of the recognition distribution is computed.

be interpreted as a recurrent coupling of the $\mathbf{v}$ units through the matrix $-\mathbf{G}^{\mathrm{T}} \cdot \mathbf{G}$. Finally, the term $g'(v_a)$ plays the same role as the term $-v_a$ that would appear in the rate equations of chapter 7. If $g'(v) \neq -v$, this can be interpreted as a modified form of firing-rate dynamics. Figure 10.6 shows the resulting network. The feedback connections from the $\mathbf{v}$ units to the input units that determine the mean of the generative distribution, $\mathbf{G} \cdot \mathbf{v}$ (equation 10.30), are also shown in this figure.

After $\mathbf{v}(\mathbf{u})$ has been determined during the E phase of EM, a delta rule (chapter 8) is used during the M phase to modify $\mathbf{G}$ and improve the generative model. The full learning rule is given in the appendix. The delta rule follows from maximizing $\mathcal{F}(\mathbf{v}(\mathbf{u}), \mathcal{G})$ with respect to $\mathbf{G}$. A complication arises here because the matrix $\mathbf{G}$ always appears multiplied by $\mathbf{v}$. This means that the bias toward small values of v_a imposed by the prior can be effectively neutralized by scaling up $\mathbf{G}$. This complication results from the approximation of deterministic recognition. To prevent the weights from growing without bound, constraints are applied on the lengths of the generative weights for each cause, $\sum_b G_{ba}^2$, to encourage the variances of all the different v_a to be approximately equal (see the appendix). Further, it is conventional to precondition the inputs before learning by whitening them so that $\langle \mathbf{u} \rangle = 0$ and $\langle \mathbf{u}\mathbf{u} \rangle = \mathbf{I}$. This typically makes learning faster, and it also ensures that the network is forced to find statistical structure beyond second order that would escape simpler methods such as factor analysis or principal components analysis. In the case that the input is created by sampling (e.g., pixelating an image), more sophisticated forms of preconditioning can be used to remove the resulting artifacts.

Applying the sparse coding model to inputs coming from the pixel intensities of small square patches of monochrome photographs of natural scenes leads to selectivities that resemble those of cortical simple cells. Before studying this result, we need to specify how the selectivities of generative models, such as the sparse coding model, are defined. The selectivities of sensory neurons are typically described by receptive fields, as in chapter 2. For a causal model, one definition of a receptive field for unit a is the set of inputs $\mathbf{u}$ for which v_a is likely to take large values. However, it may be

projective field

impossible to construct receptive fields by averaging over these inputs in nonlinear models, such as sparse coding models. Furthermore, generative models are most naturally characterized by projective fields rather than receptive fields. The projective field associated with a particular cause v_a can be defined as the set of inputs that it frequently generates. This consists of all the $\mathbf{u}$ values for which $P[\mathbf{u}|v_a; \mathcal{G}]$ is sufficiently large when v_a is large. For the model of figure 10.1, the projective fields are simply the circles in figure 10.1C. It is important to remember that projective fields can be quite different from receptive fields.

Projective fields for the Olshausen and Field model trained on natural scenes are shown in figure 10.7A, with one picture for each component of $\mathbf{v}$. In this case, the projective field for v_a is simply the matrix elements G_{ab} plotted for all b values. In figure 10.7A, the index b is plotted over a two-dimensional grid representing the location of the input u_b within the visual field. The projective fields form a Gabor-like representation for images, covering a variety of spatial scales and orientations. The resemblance of this representation to the receptive fields of simple cells in primary visual cortex is quite striking, although these are the projective fields of the model, not its receptive fields. Unfortunately, there is no simple form for the receptive fields of the $\mathbf{v}$ units. Figure 10.7B compares the projective field of one unit with receptive fields determined by presenting either dots or gratings as inputs and recording the responses. The responses to the dots directly determine the receptive field, while responses to the gratings directly determine the Fourier transform of the receptive field. Differences between the receptive fields calculated on the basis of these two types of input are evident in the figure. In particular, the receptive field computed from gratings shows more spatial structure than the one mapped by dots. Nevertheless, both show a resemblance to the projective field and to a typical simple-cell receptive field.

In a generative model, projective fields are associated with the causes underlying the visual images presented during training. The fact that the causes extracted by the sparse coding model resemble Gabor patches within the visual field is somewhat strange from this perspective. It is difficult to conceive of images as arising from such low-level causes, instead of causes couched in terms of objects within the images, for example. From the perspective of good representation, causes that are more like objects and less like Gabor patches would be more useful. To put this another way, although the prior distribution over causes biased them toward mutual independence, the causes produced by the recognition model in response to natural images are not actually independent. This is due to the structure in images arising from more complex objects than bars and gratings. It is unlikely that this high-order structure can be extracted by a model with only one set of causes. It is more natural to think of causes in a hierarchical manner, with causes at a higher level accounting for structure in the causes at a lower level. The multiple representations in areas along the visual pathway suggests such a hierarchical scheme, but the corresponding models are still in the rudimentary stages of development.

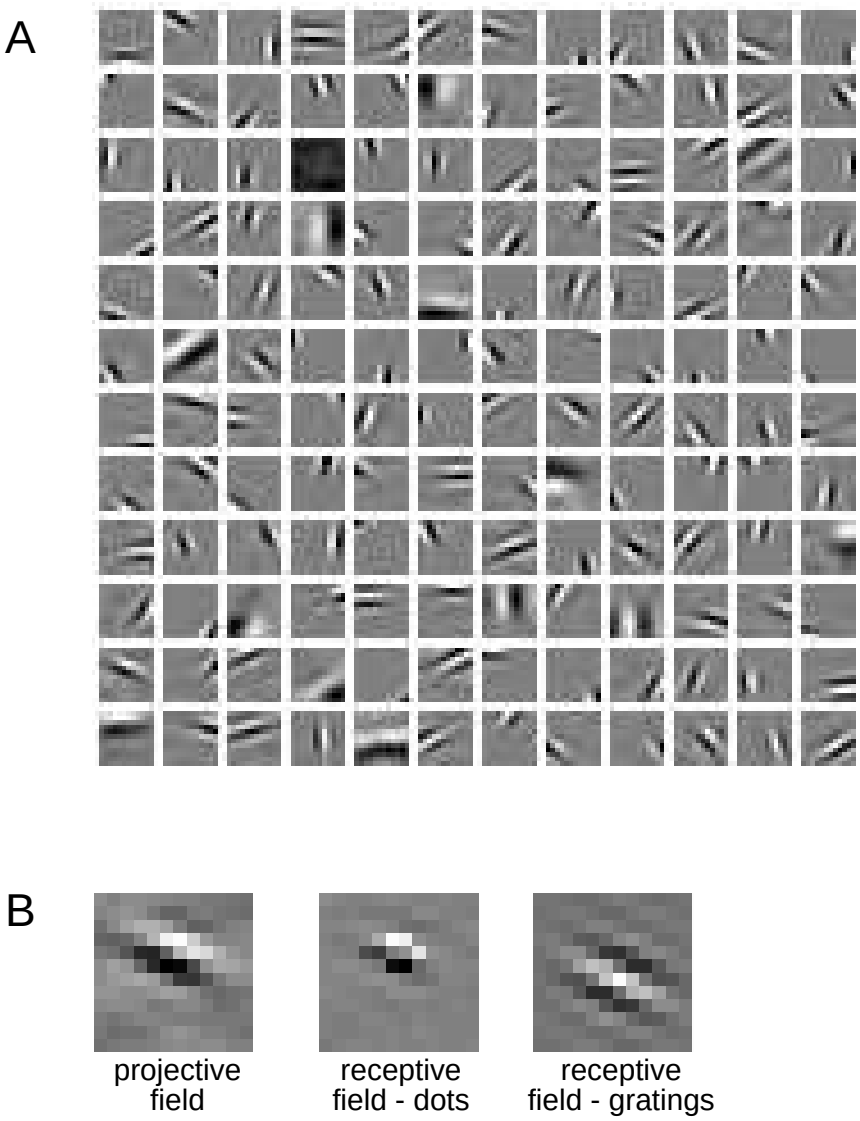

Figure 10.7 Projective and receptive fields for a sparse coding network with $N_u = N_v = 144$. (A) Projective fields $\mathbf{G}_{ab}$ with a indexing representational units (the components of $\mathbf{v}$), and b indexing input units $\mathbf{u}$ on a 12×12 pixel grid. Each box represents a different a value, and the b values are represented within the box by the corresponding input location. Weights are represented by the gray-scale level, with gray indicating 0. (B) The relationship between projective and receptive fields. The left panel shows the projective field of one of the units in A. The middle and right panels show its receptive field mapped using inputs generated by dots and by gratings, respectively. (Adapted from Olshausen & Field, 1997.)

Independent Components Analysis

As for the case of the mixtures of Gaussians model and factor analysis, an interesting model emerges from sparse coding as $\Sigma \to 0$. In this limit, the generative distribution (equation 10.30) approaches a δ function and always generates $\mathbf{u}(\mathbf{v}) = \mathbf{G} \cdot \mathbf{v}$. Under the additional restriction that there are as many causes as inputs, the approximation we used for the sparse coding model of making the recognition distribution deterministic becomes exact, and the recognition distribution that maximizes $\mathcal{F}$ is

$$Q[\mathbf{v}; \mathbf{u}] = |\det \mathbf{W}|^{-1} \delta(\mathbf{u} - \mathbf{W}^{-1} \cdot \mathbf{v}), \tag{10.35}$$

where $\mathbf{W} = \mathbf{G}^{-1}$ is the matrix inverse of the generative weight matrix. The factor $|\det \mathbf{W}|$ comes from the normalization condition on Q, $\int d\mathbf{v}\, Q(\mathbf{v}; \mathbf{u}) = 1$. At the maximum with respect to Q, the function $\mathcal{F}$ is

$$\mathcal{F}(Q, G) = \left\langle -\frac{1}{2\Sigma} |\mathbf{u} - \mathbf{G} \cdot \mathbf{W} \cdot \mathbf{u}|^2 + \sum_a g([\mathbf{W} \cdot \mathbf{u}]_a) \right\rangle + \ln |\det \mathbf{W}| + K, \tag{10.36}$$

where K is independent of $\mathbf{G}$. Under the conventional EM procedure, we would maximize this expression with respect to $\mathbf{G}$, keeping $\mathbf{W}$ fixed. However, the normal procedure fails in this case, because the minimum of the right side of equation 10.36 occurs at $\mathbf{G} = \mathbf{W}^{-1}$, and $\mathbf{W}$ is being held fixed, so $\mathbf{G}$ cannot change. This is an anomaly of coordinate ascent in this particular limit.

Fortunately, it is easy to fix this problem, because we know that $\mathbf{W} = \mathbf{G}^{-1}$ provides an exact inversion of the generative model. Therefore, instead of holding $\mathbf{W}$ fixed during the M phase of an EM procedure, we keep $\mathbf{W} = \mathbf{G}^{-1}$ at all times as we change $\mathbf{G}$. This sets $\mathcal{F}$ equal to the average log likelihood, and the process of optimizing with respect to $\mathbf{G}$ is equivalent to likelihood maximization. Because $\mathbf{W} = \mathbf{G}^{-1}$, maximizing with respect to $\mathbf{W}$ is equivalent to maximizing with respect to $\mathbf{G}$, and it turns out that this is easier to do. Therefore, we set $\mathbf{W} = \mathbf{G}^{-1}$ in equation 10.36, which causes the first term to vanish, and write the remaining terms as the log likelihood expressed as a function of $\mathbf{W}$ instead of $\mathbf{G}$,

$$L(\mathbf{W}) = \left\langle \sum_a g([\mathbf{W} \cdot \mathbf{u}]_a) \right\rangle + \ln |\det \mathbf{W}| + K. \tag{10.37}$$

Direct stochastic gradient ascent on this log likelihood can be performed using the update rule

$$\mathbf{W}_{ab} \to \mathbf{W}_{ab} + \epsilon \left([\mathbf{W}^{-1}]_{ba} + g'(v_a) u_b \right), \tag{10.38}$$

where ϵ is a small learning rate parameter, and we have used the fact that $\partial \ln |\det \mathbf{W}| / \partial \mathbf{W}_{ab} = [\mathbf{W}^{-1}]_{ba}$.

The update rule of equation 10.38 can be simplified by using a clever trick. Because $\mathbf{W}^T \mathbf{W}$ is a positive definite matrix (see the Mathematical

Appendix), the weight change can be multiplied by $\mathbf{W}^T\mathbf{W}$ without affecting the fixed points of the update rule. This means that the alternative learning rule

$$\mathbf{W}_{ab} \rightarrow \mathbf{W}_{ab} + \epsilon \left(\mathbf{W}_{ab} + g'(v_a)\,[\mathbf{v} \cdot \mathbf{W}]_b\right) \tag{10.39}$$

has the same potential final weight matrices as equation 10.38. This is called a natural gradient rule, and it avoids the matrix inversion of $\mathbf{W}$ as well as providing faster convergence. Equation 10.39 can be interpreted as the sum of an anti-decay term that forces $\mathbf{W}$ away from 0, and a generalized type of anti-Hebbian term. The choice of prior $p[v] \propto 1/\cosh(v)$ makes $g'(v) = -\tanh(v)$ and produces the rule

$$\mathbf{W}_{ab} \rightarrow \mathbf{W}_{ab} + \epsilon \left(\mathbf{W}_{ab} - \tanh(v_a)\,[\mathbf{v} \cdot \mathbf{W}]_b\right) . \tag{10.40}$$

This algorithm is called independent components analysis. Just as the sparse coding network is a nonlinear generalization of factor analysis, independent components analysis is a nonlinear generalization of principal components analysis that attempts to account for non-Gaussian features of the input distribution. The generative model is based on the assumption that $\mathbf{u} = \mathbf{G} \cdot \mathbf{v}$. Some other technical conditions must be satisfied for independent components analysis to extract reasonable causes. Specifically, the prior distributions over causes $p[v] \propto \exp(g(v))$ must be non-Gaussian and, at least to the extent of being correctly super- or sub-Gaussian, must faithfully reflect the actual distribution over causes. The particular form $p[v] \propto 1/\cosh(v)$ is super-Gaussian, and thus generates a sparse prior. There are variants of independent components analysis in which the prior distributions are adaptive.

The independent components algorithm was suggested by Bell and Sejnowski (1995) from the different perspective of maximizing the mutual information between $\mathbf{u}$ and $\mathbf{v}$ when $v_a(\mathbf{u}) = f([\mathbf{W} \cdot \mathbf{u}]_a)$, with a particular, monotonically increasing nonlinear function f. Maximizing the mutual information in this context requires maximizing the entropy of the distribution over $\mathbf{v}$. This, in turn, requires the components of $\mathbf{v}$ to be as independent as possible because redundancy between them reduces the entropy. In the case that $f(v) = g'(v)$, the expression for the entropy is the same as that for the log likelihood in equation 10.37, up to constant factors. Thus, maximizing the entropy and performing maximum likelihood density estimation are identical.

One advantage independent components analysis has over other sparse coding algorithms is that, because the recognition model is an exact inverse of the generative model, receptive as well as projective fields can be constructed. Just as the projective field for v_a can be represented by the matrix elements G_{ab} for all b values, the receptive field is given by W_{ab} for all b.

To illustrate independent components analysis, figure 10.8 shows an (admittedly bizarre) example of its application to the sounds created by tapping a tooth while adjusting the shape of the mouth to reproduce a

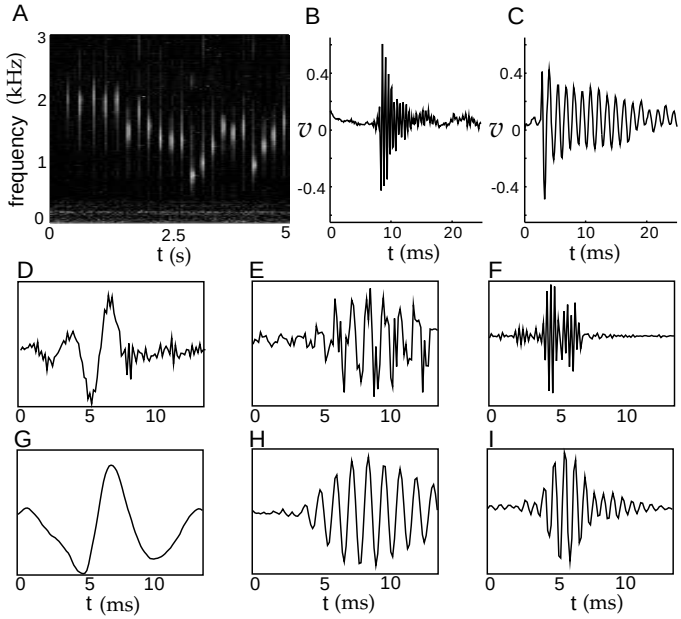

Figure 10.8 Independent components of tooth-tapping sounds. (A) Spectrogram of the input. (B-C) Waveforms for high- and low-frequency notes. The mouth acts as a damped resonant cavity in the generation of these tones. (D-F) Three independent components calculated on the basis of $1/80$ s samples taken from the input at random times. The graphs show the receptive fields (from $\mathbf{W}$) for three output units. D is reported to be sensitive to the sound of an air conditioner. E and F extract tooth taps of different frequencies. (G-I) The associated projective fields (from $\mathbf{G}$), showing the input activity associated with the causes in D-F. (Adapted from Bell & Sejnowski, 1996.)

tune by Beethoven. The input, sampled at 8 kHz, has the spectrogram shown in figure 10.8A. In this example, we have some idea about likely causes. For example, the plots in figures 10.8B and 10.8C show high- and low-frequency tooth taps, although other causes arise from the imperfect recording conditions (e.g., the background sound of an air conditioner). A close variant of the independent components analysis method described above was used to extract $N_v = 100$ independent components. Figure 10.8D, 10.8E, and 10.8F show the receptive fields of three of these components. The last two extract particular frequencies in the input. Figure 10.8G, 10.8H, and 10.8I show projective fields. Note that the projective fields are much smoother than the receptive fields.

Bell and Sejnowski (1997) also used visual input data similar to those used in the example of figure 10.7, along with the prior $p[v] \propto 1/\cosh(v)$, and found that independent components analysis extracts Gabor-like receptive fields similar to the projective fields shown in figure 10.7A.

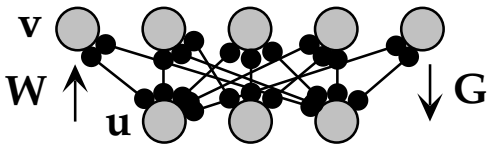

Figure 10.9 Network for the Helmholtz machine. In the bottom-up network, representational units **v** are driven by inputs **u** through feedforward weights **W**. In the top-down network, the inputs are driven by the **v** units through feedback weights **G**.

The Helmholtz Machine

The Helmholtz machine was designed to accommodate hierarchical architectures that construct complex multilayer representations. The model involves two interacting networks, one with parameters $\mathcal{G}$ that is driven in the top-down direction to implement the generative model, and the other, with parameters $\mathcal{W}$, driven bottom-up to implement the recognition model. The parameters are determined by a modified EM algorithm that results in roughly symmetric updates for the two networks.

We consider a simple, two-layer, nonlinear Helmholtz machine (figure 10.9) with binary units, so that u_b and v_a for all b and a take the values 0 or 1. For this model,

$$P[\mathbf{v};\mathcal{G}] = \prod_a \left(f(g_a)\right)^{v_a}\left(1-f(g_a)\right)^{1-v_a} \tag{10.41}$$

$$P[\mathbf{u}|\mathbf{v};\mathcal{G}] = \prod_b \left(f\left(h_b+[\mathbf{G}\cdot\mathbf{v}]_b\right)\right)^{u_b}\left(1-f\left(h_b+[\mathbf{G}\cdot\mathbf{v}]_b\right)\right)^{1-u_b}, \tag{10.42}$$

where g_a is a generative bias weight for output a that controls how frequently $v_a = 1$, h_b is the generative bias weight for u_b, and $f(g) = 1/(1+\exp(-g))$ is the standard sigmoid function. The generative model is thus parameterized by $\mathcal{G} = (\mathbf{g}, \mathbf{h}, \mathbf{G})$. According to these distributions, the components of **v** are mutually independent, and the components of **u** are independent given a fixed value of **v**.

The generative model is noninvertible in this case, so an approximate recognition distribution must be constructed. This uses a form similar to equation 10.42, except with bottom-up weights **W** and biases **w**,

$$Q[\mathbf{v};\mathbf{u},\mathcal{W}] = \prod_a \left(f\left(w_a+[\mathbf{W}\cdot\mathbf{u}]_a\right)\right)^{v_a}\left(1-f\left(w_a+[\mathbf{W}\cdot\mathbf{u}]_a\right)\right)^{1-v_a} \tag{10.43}$$

The parameter list for the recognition model is $\mathcal{W} = (\mathbf{w}, \mathbf{W})$. This distribution is only an approximate inverse of the generative model because it implies that the components of **v** are independent when, in fact, given a particular input **u**, they are conditionally dependent due to the way they interact in equation 10.42 to generate **u** (this is the same assumption as in the mean-field approximate distribution for the Boltzmann machine, except that the parameters of the distribution here are shared between all input cases).

The EM algorithm for this noninvertible model would consist of alternately maximizing the function $\mathcal{F}$ given by

$$\mathcal{F}(\mathcal{W}, \mathcal{G}) = \left\langle \sum_{\mathbf{v}} Q[\mathbf{v}; \mathbf{u}, \mathcal{W}] \ln \frac{P[\mathbf{v}, \mathbf{u}; \mathcal{G}]}{Q[\mathbf{v}; \mathbf{u}, \mathcal{W}]} \right\rangle \tag{10.44}$$

with respect to the parameters $\mathcal{W}$ and $\mathcal{G}$. For the M phase of the Helmholtz machine, this is exactly what is done. However, during the E phase, maximizing with respect to $\mathcal{W}$ is problematic because the function $Q[\mathbf{v}; \mathbf{u}, \mathcal{W}]$ appears in two places in the expression for $\mathcal{F}$. This also makes the learning rule during the E phase take a different form from that during the M phase. Instead, the Helmholtz machine uses a simpler and more symmetric approximation to EM.

The approximation to EM used by the Helmholtz machine is constructed by re-expressing $\mathcal{F}$ from equation 10.10, explicitly writing out the average over input data and the expression for the Kullback-Leibler divergence,

$$\begin{aligned}\mathcal{F}(\mathcal{W}, \mathcal{G}) &= L(\mathcal{G}) - \sum_{\mathbf{u}} P[\mathbf{u}] D_{\mathrm{KL}}(Q[\mathbf{v}; \mathbf{u}, \mathcal{W}], P[\mathbf{v}|\mathbf{u}; \mathcal{G}]) \\ &= L(\mathcal{G}) - \sum_{\mathbf{u}} P[\mathbf{u}] \sum_{\mathbf{v}} Q[\mathbf{v}; \mathbf{u}, \mathcal{W}] \ln \left(\frac{Q[\mathbf{v}; \mathbf{u}, \mathcal{W}]}{P[\mathbf{v}|\mathbf{u}; \mathcal{G}]} \right) . \end{aligned} \tag{10.45}$$

This is the function that is maximized with respect to $\mathcal{G}$ during the M phase for the Helmholtz machine. However, the E phase is not based on maximizing equation 10.45 with respect to $\mathcal{W}$. Instead, an approximate $\mathcal{F}$ function that we call $\tilde{\mathcal{F}}$ is used. This is constructed by using $P[\mathbf{u}; \mathcal{G}]$ as an approximation for $P[\mathbf{u}]$ and $D_{\mathrm{KL}}(P[\mathbf{v}|\mathbf{u}; \mathcal{G}], Q[\mathbf{v}; \mathbf{u}, \mathcal{W}])$ as an approximation for $D_{\mathrm{KL}}(Q[\mathbf{v}; \mathbf{u}, \mathcal{W}], P[\mathbf{v}|\mathbf{u}; \mathcal{G}])$ in equation 10.45. These are likely to be good approximations if the generative and approximate recognition models are accurate. Thus, we write

$$\begin{aligned}\tilde{\mathcal{F}}(\mathcal{W}, \mathcal{G}) &= L(\mathcal{G}) - \sum_{\mathbf{u}} P[\mathbf{u}; \mathcal{G}] D_{\mathrm{KL}}(P[\mathbf{v}|\mathbf{u}; \mathcal{G}], Q[\mathbf{v}; \mathbf{u}, \mathcal{W}]) \\ &= L(\mathcal{G}) - \sum_{\mathbf{u}} P[\mathbf{u}; \mathcal{G}] \sum_{\mathbf{v}} P[\mathbf{v}|\mathbf{u}; \mathcal{G}] \ln \left(\frac{P[\mathbf{v}|\mathbf{u}; \mathcal{G}]}{Q[\mathbf{v}; \mathbf{u}, \mathcal{W}]} \right) . \end{aligned} \tag{10.46}$$

and maximize this, rather than $\mathcal{F}$, with respect to $\mathcal{W}$ during the E phase. This amounts to averaging the "flipped" Kullback-Leibler divergence over samples of $\mathbf{u}$ created by the generative model, rather than real data samples. The advantage of making these approximations is that the E and M phases become highly symmetric, as can be seen by examining the second equalities in equations 10.45 and 10.46.

Learning in the Helmholtz machine proceeds by using stochastic sampling to replace the weighted sums in equations 10.45 and 10.46. In the M phase, an input $\mathbf{u}$ from $P[\mathbf{u}]$ is presented, and a sample $\mathbf{v}$ is drawn from the current recognition distribution $Q[\mathbf{v}; \mathbf{u}, \mathcal{W}]$. Then the generative weights $\mathcal{G}$ are changed according to the discrepancy between $\mathbf{u}$ and the generative or top-down prediction $\mathbf{f}(\mathbf{h} + \mathbf{G} \cdot \mathbf{v})$ of $\mathbf{u}$ (see the appendix). Thus, the generative model is trained to make $\mathbf{u}$ more likely to be generated by the cause

v associated with it by the recognition model. In the E phase, samples of both **v** and **u** are drawn from the generative model distributions $P[\mathbf{v}; \mathcal{G}]$ and $P[\mathbf{u}|\mathbf{v}; \mathcal{G}]$, and the recognition parameters $\mathcal{W}$ are changed according to the discrepancy between the sampled cause **v** and the recognition or bottom-up prediction $\mathbf{f}(\mathbf{w} + \mathbf{W} \cdot \mathbf{u})$ of **v** (see the appendix). The rationale for this is that the **v** that was used by the generative model to create **u** is a good choice for its cause in the recognition model.

wake-sleep algorithm

The two phases of learning are sometimes called wake and sleep because learning in the first phase is driven by real inputs **u** from the environment, while learning in the second phase is driven by values **v** and **u** "fantasized" by the generative model. This terminology is based on slightly different principles from the wake and sleep phases of the Boltzmann machine discussed in chapter 8. The sleep phase is only an approximation of the actual E phase, and general conditions under which learning converges appropriately are not known.

10.4 Discussion

Because of the widespread significance of coding, transmitting, storing, and decoding visual images such as photographs and movies, substantial effort has been devoted to understanding the structure of this class of inputs. As a result, visual images provide an ideal testing ground for representational learning algorithms, allowing us to go beyond evaluating the representations they produce solely in terms of the log likelihood and qualitative similarities with cortical receptive fields.

Most modern image (and auditory) processing techniques are based on multi-resolution decompositions. In such decompositions, images are represented by the activity of a population of units with systematically varying spatial frequency and orientation preferences, centered at various locations on the image. The outputs of the representational units are generated by filters (typically linear) that act as receptive fields and are partially localized in both space and spatial frequency. The filters usually have similar underlying forms, but they are cast at different spatial scales and centered at different locations for the different units. Systematic versions of such representations, in forms such as wavelets, are important signal processing tools, and there is an extensive body of theory about their representational and coding qualities. Representation of sensory information in separated frequency bands at different spatial locations has significant psychophysical consequences as well.

The projective fields of the units in the sparse coding network shown in figure 10.7 suggest that they construct something like a multi-resolution decomposition of inputs, with multiple spatial scales, locations, and orientations. Thus, multi-resolution analysis gives us a way to put into sharper focus the issues arising from models such as sparse coding and independent components analysis. After a brief review of multi-resolution decom-

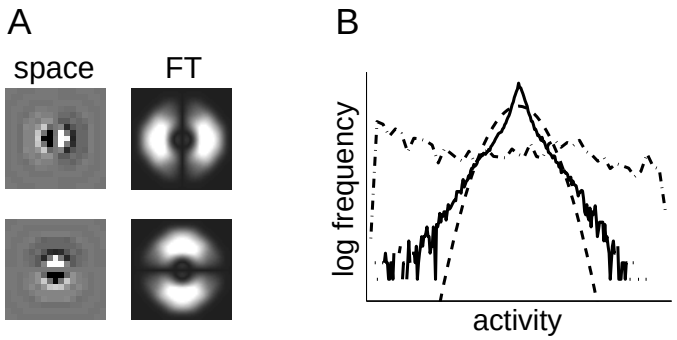

Figure 10.10 Multi-resolution filtering. (A) Vertical and horizontal filters (left) and their Fourier transforms (right) that are used at multiple positions and spatial scales to generate a multi-resolution representation. The rows of the matrix $\mathbf{W}$ are displayed here in gray scale on a two-dimensional grid representing the location of the corresponding input. (B) Log frequency distribution of the outputs of the highest spatial frequency filters (solid line) compared with a Gaussian distribution with the same mean and variance (dashed line) and the distribution of pixel values for the image shown in figure 10.11A (dot-dashed line). The pixel values of the image were rescaled to fit into the range. (Adapted from Simoncelli & Freeman, 1995; Karasaridis & Simoncelli, 1996.)

positions, we use them to consider various properties of representational learning from the perspective of information transmission and sparseness, overcompleteness, and residual dependencies between inferred causes.

Multi-resolution Decomposition

Many multi-resolution decompositions, with a variety of computational and representational properties, can be expressed as linear transformations $\mathbf{v} = \mathbf{W} \cdot \mathbf{u}$, where the rows of $\mathbf{W}$ describe filters, such as those illustrated in figure 10.10A. Figure 10.11 shows the result of applying multi-resolution filters, constructed by scaling and shifting the filters shown in figure 10.10A, to the photograph in figure 10.11A. Vertical and horizontal filters similar to those in figure 10.10A, but with different sizes, produce the decomposition shown in figures 10.11B-10.11D and 10.11F-10.11H when translated across the image. The level of gray indicates the output generated by placing the different filters over the corresponding point on the image. These outputs, plus the low-pass image in figure 10.11E and an extra high-pass image that is not shown, can be used to reconstruct the whole photograph almost perfectly through a generative process that is the inverse of the recognition process.

Coding

One reason for using multi-resolution decompositions is that they offer efficient ways of encoding visual images, whereas raw values of input pixels

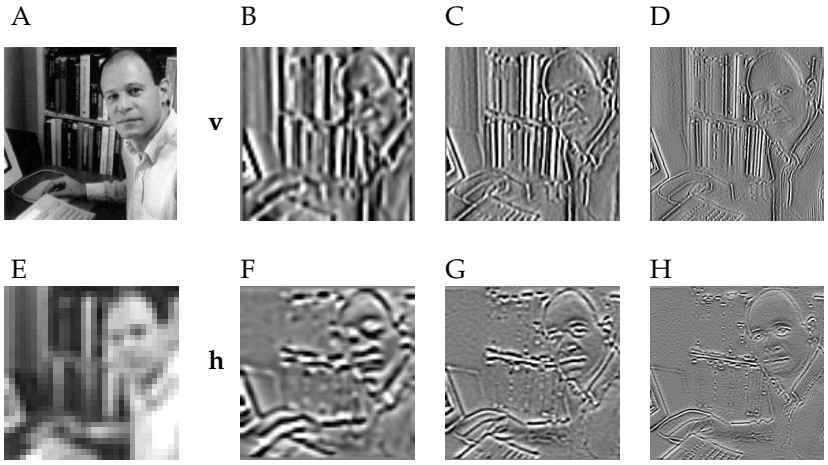

Figure 10.11 Multi-resolution image decomposition. A gray-scale image is decomposed, using the pair of vertical and horizontal filters shown in figure 10.10. (A) The original image. (B-D) The outputs of successively higher spatial frequency, vertically oriented filters translated across the image. (E) The image after passage through a low-pass filter. (F-H) The outputs of successively higher spatial frequency, horizontally oriented filters translated across the image.

provide an inefficient encoding. This is illustrated by the dot-dashed line in figure 10.10B, which shows that the distribution over the values of the input pixels of the image in figure 10.11A is approximately flat or uniform. Up to the usual additive constants related to the precision with which filter outputs are encoded, the contribution to the coding cost from a single unit is the entropy of the probability distribution of its output. The distribution over pixel intensities is flat, which is the maximum entropy distribution for a variable with a fixed range. Encoding the individual pixel values therefore incurs the maximum possible coding cost.

By contrast, the solid line in figure 10.10B shows the distribution of the outputs of the finest scale vertically and horizontally tuned filters (figures 10.11D and 10.11H) in response to figure 10.11A. The filter outputs have a sparse distribution similar to the double exponential distribution in figure 10.5B. This distribution has significantly lower entropy than the uniform distribution, so the filter outputs provide a more efficient encoding than pixel values.

In making these statements about the distributions of activities, we are equating the output distribution of a filter applied at many locations on a single image with the output distribution of a filter applied at a fixed location on many images. This assumes spatial translational invariance of the ensemble of visual images.

Images represented by multi-resolution filters can be further compressed by retaining only approximate values of the filter outputs. This is called lossy coding and may consist of reporting filter outputs as integer multiples of a basic unit. Making the multi-resolution code for an image lossy

lossy coding

by coarsely quantizing the outputs of the highest spatial frequency filters generally has quite minimal perceptual consequences, while saving substantial coding cost (because these outputs are most numerous). This fact illustrates the important point that trying to build generative models of all aspects of visual images may be unnecessarily difficult, because only certain aspects of images are actually relevant. Unfortunately, abstract principles are unlikely to tell us what information in the input can safely be discarded independent of details of how the representations are to be used.

Overcomplete Representations

Sparse representations often have more output units than input units. Such representations, called overcomplete, are the subject of substantial work in multi-resolution theory. Many reasons have been suggested for overcompleteness, although none obviously emerges from the requirement of fitting good probabilistic models to input data.

One interesting idea comes from the notion that the task of manipulating representations should be invariant to the groups of symmetry transformations of the input, which, for images, include rotation, translation, and scaling. Complete representations are minimal, and so do not densely sample orientations. This means that the operations required to manipulate images of objects presented at angles not directly represented by the filters are different from those required at the represented angles (such as horizontal and vertical for the example of figure 10.10). When a representation is overcomplete in such a way that different orientations are represented roughly equally, as in primary visual cortex, the computational operations required to manipulate images are more uniform as a function of image orientation. Similar ideas apply across scale, so that the operations required to manipulate large and small images of the same object (as if viewed from near and far) are likewise similar. However, it is impossible to generate representations that satisfy all these constraints perfectly.

In more realistic models that include noise, other rationales for overcompleteness come from considering population codes, in which many units redundantly report information about closely related quantities so that uncertainty can be reduced. Despite the ubiquity of overcomplete population codes in the brain, there are few representational learning models that produce them satisfactorily. The coordinated representations required to construct population codes are often incompatible with other heuristics such as factorial or sparse coding.

Interdependent Causes

One of the failings of multi-resolution decompositions for coding is that the outputs are not mutually independent. This makes encoding each of

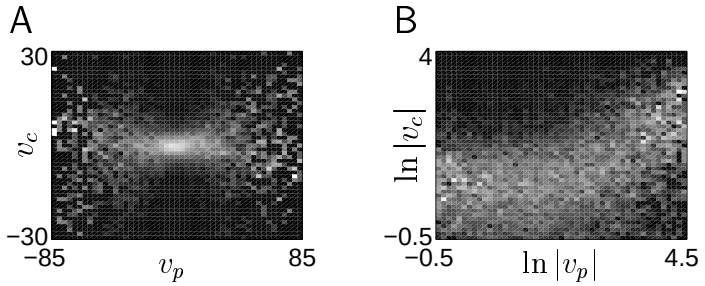

Figure 10.12 (A) Gray-scale plot of the conditional distribution of the output of a filter at the finest spatial scale (v_c) given the output of a coarser filter (v_p) with the same position and orientation (using the picture in figure 10.11A as input data). Each column is separately normalized. The plot has a characteristic bow-tie shape. (B) The same data plotted as the conditional distribution of $\ln|v_c|$ given $\ln|v_p|$. (Adapted from Simoncelli & Adelson, 1990; Simoncelli & Schwartz, 1999.)

the redundant filter outputs wasteful. Figure 10.12 illustrates such an interdependence by showing the conditional distribution for the output v_c of a horizontally tuned filter at a fine scale, given the output v_p of a horizontally tuned unit at the next coarser scale. The plots show gray-scale values of the conditional probability density $p[v_c|v_p]$. The mean of this distribution is roughly 0, but there is a clear correlation between the magnitude of $|v_p|$ and the variance of v_c. This means that structure in the image is coordinated across different spatial scales, so that high outputs from a coarse scale filter are typically accompanied by substantial output (of one sign or the other) at a finer scale. Following Simoncelli (1997), we plot the conditional distribution of $\ln|v_c|$ given $\ln|v_p|$ in figure 10.12B. For small values of $\ln|v_p|$, the distribution of $\ln|v_c|$ is flat, but for larger values of $\ln|v_p|$ the growth in the value of $|v_c|$ is clear.

The interdependence shown in figure 10.12 suggests a failing of sparse coding to which we have alluded before. Although the prior distribution for sparse coding stipulates independent causes, the causes identified as underlying real images are not independent. The dependence apparent in figure 10.12 can be removed by a nonlinear transformation in which the outputs of the units normalize each other (similar to the model introduced to explain contrast saturation in chapter 2). This transformation can lead to more compact codes for images. However, the general problem suggests that something is amiss with the heuristic of seeking independent causes for representations early in the visual pathway.

The most important dependencies as far as causal models are concerned are those induced by the presence in images of objects with large-scale coordinated structure. Finding and building models of these dependencies is the goal for more sophisticated, hierarchical representational learning schemes aimed ultimately at object recognition within complex visual scenes.

10.5 Chapter Summary

We have presented a systematic treatment of exact and approximate maximum likelihood density estimation as a way of fitting probabilistic generative models and thereby performing representational learning. Recognition models, which are the statistical inverses of generative models, specify the causes underlying an input and play a crucial role in learning. We discussed the expectation maximization (EM) algorithm applied to invertible and noninvertible models, including the use of deterministic and probabilistic approximate recognition models and a lower bound on the log likelihood.

We presented a variety of models for continuous inputs with discrete, continuous, or vector-valued causes. These include mixture of Gaussians, K-means, factor analysis, principal components analysis, sparse coding, and independent components analysis. We also described the Helmholtz machine and discussed general issues of multi-resolution representation and coding.

10.6 Appendix

Summary of Causal Models

Model	Generative Model	Recognition Model	Learning Rules
mixture of Gaussians	$P[v;\mathcal{G}] = \gamma_v$ $P[\mathbf{u}\|v;\mathcal{G}] = \mathcal{N}(\mathbf{u};\mathbf{g}_v,\Sigma_v)$	$P[v\|\mathbf{u};\mathcal{G}] \propto \gamma_v \mathcal{N}(\mathbf{u};\mathbf{g}_v,\Sigma_v)$	$\mu_v = \langle P[v\|\mathbf{u};\mathcal{G}]\rangle$ $\mathbf{g}_v = \langle P[v\|\mathbf{u};\mathcal{G}]\mathbf{u}\rangle/\gamma_v$ $\Sigma_v = \langle P[v\|\mathbf{u};\mathcal{G}]\|\mathbf{u}-\mathbf{g}_v\|^2\rangle/(N_u\gamma_v)$
factor analysis	$P[\mathbf{v};\mathcal{G}] = \mathcal{N}(\mathbf{v};\mathbf{0},1)$ $P[\mathbf{u}\|\mathbf{v};\mathcal{G}] = \mathcal{N}(\mathbf{u};\mathbf{G}\cdot\mathbf{v},\mathbf{\Sigma})$ $\mathbf{\Sigma} = \mathrm{diag}\left(\Sigma_1,\Sigma_2,\ldots,\Sigma_{N_u}\right)$	$P[\mathbf{v}\|\mathbf{u};\mathcal{G}] = \mathcal{N}(\mathbf{v};\mathbf{W}\cdot\mathbf{u},\mathbf{\Psi})$ $\mathbf{\Psi} = \left(\mathbf{I}+\mathbf{G}^{\mathrm{T}}\cdot\mathbf{\Sigma}^{-1}\cdot\mathbf{G}\right)^{-1}$ $\mathbf{W} = \mathbf{\Psi}\cdot\mathbf{G}^{\mathrm{T}}\cdot\mathbf{\Sigma}^{-1}$	$\mathbf{G} = \mathbf{C}\cdot\mathbf{W}^{\mathrm{T}}\cdot(\mathbf{W}\cdot\mathbf{C}\cdot\mathbf{W}^{\mathrm{T}}+\mathbf{\Psi})^{-1}$ $\mathbf{\Sigma} = \mathrm{diag}\left(\mathbf{G}\cdot\mathbf{\Psi}\cdot\mathbf{G}^{\mathrm{T}} + (\mathbf{I}-\mathbf{G}\cdot\mathbf{W})\cdot\mathbf{C}\cdot(\mathbf{I}-\mathbf{G}\cdot\mathbf{W})^{\mathrm{T}}\right)$ $\mathbf{C} = \langle\mathbf{u}\mathbf{u}\rangle$
principal components analysis	$P[\mathbf{v};\mathcal{G}] = \mathcal{N}(\mathbf{v};\mathbf{0},1)$ $\mathbf{u} = \mathbf{G}\cdot\mathbf{v}$	$\mathbf{v} = \mathbf{W}\cdot\mathbf{u}$ $\mathbf{W} = (\mathbf{G}^{\mathrm{T}}\cdot\mathbf{G})^{-1}\cdot\mathbf{G}^{\mathrm{T}}$	$\mathbf{G} = \mathbf{C}\cdot\mathbf{W}^{\mathrm{T}}\cdot(\mathbf{W}\cdot\mathbf{C}\cdot\mathbf{W}^{\mathrm{T}})^{-1}$ $\mathbf{C} = \langle\mathbf{u}\mathbf{u}\rangle$
sparse coding	$P[\mathbf{v};\mathcal{G}] \propto \prod_a \exp(g(v_a))$ $P[\mathbf{u}\|\mathbf{v};\mathcal{G}] = \mathcal{N}(\mathbf{u};\mathbf{G}\cdot\mathbf{v},\Sigma)$	$\mathbf{G}^{\mathrm{T}}\cdot(\mathbf{u}-\mathbf{G}\cdot\mathbf{v}) + \mathbf{g}'(\mathbf{v}) = 0$	$\mathbf{G} \to \mathbf{G} + \epsilon(\mathbf{u}-\mathbf{G}\cdot\mathbf{v})\mathbf{v}$ $\left(\sum_b G_{ba}^2\right) \to \left(\sum_b G_{ba}^2\right)\left(\langle v_a^2\rangle - \langle v_a\rangle^2/\sigma^2\right)^{0.01}$
independent components analysis	$P[\mathbf{v};\mathcal{G}] \propto \prod_a \exp(g(v_a))$ $\mathbf{u} = \mathbf{G}\cdot\mathbf{v}$	$\mathbf{v} = \mathbf{W}\cdot\mathbf{u}$ $\mathbf{W} = \mathbf{G}^{-1}$	$\mathbf{W}_{ab} \to \mathbf{W}_{ab} + \epsilon\left(\mathbf{W}_{ab} + g'(v_a)[\mathbf{v}\cdot\mathbf{W}]_b\right)$ $g'(v) = -\tanh(v)$ if $g(v) = -\ln\cosh(v)$
binary Helmholtz machine	$P[\mathbf{v};\mathcal{G}] = \prod_a \left(f(g_a)\right)^{v_a}\left(1-f(g_a)\right)^{1-v_a}$ $P[\mathbf{u}\|\mathbf{v};\mathcal{G}] = \prod_b \left(f_b(\mathbf{h}+\mathbf{G}\cdot\mathbf{v})\right)^{u_b} \times \left(1-f_b(\mathbf{h}+\mathbf{G}\cdot\mathbf{v})\right)^{1-u_b}$ $f_b(\mathbf{h}+\mathbf{G}\cdot\mathbf{v}) = f\left(h_b + [\mathbf{G}\cdot\mathbf{v}]_b\right)$	$Q[\mathbf{v};\mathbf{u},\mathcal{W}] = \prod_a \left(f_a(\mathbf{w}+\mathbf{W}\cdot\mathbf{u})\right)^{v_a} \times \left(1-f_a(\mathbf{w}+\mathbf{W}\cdot\mathbf{u})\right)^{1-v_a}$ $f_a(\mathbf{w}+\mathbf{W}\cdot\mathbf{u}) = f\left(w_a + [\mathbf{W}\cdot\mathbf{u}]_a\right)$	wake: $\mathbf{u} \sim P[\mathbf{u}]$, $\mathbf{v} \sim Q[\mathbf{v};\mathbf{u},\mathcal{W}]$ $\mathbf{g} \to \mathbf{g} + \epsilon(\mathbf{v}-\mathbf{f}(\mathbf{g}))$ $\mathbf{h} \to \mathbf{h} + \epsilon(\mathbf{u}-\mathbf{f}(\mathbf{h}+\mathbf{G}\cdot\mathbf{v}))$ $\mathbf{G} \to \mathbf{G} + \epsilon(\mathbf{u}-\mathbf{f}(\mathbf{h}+\mathbf{G}\cdot\mathbf{v}))\mathbf{v}$ sleep: $\mathbf{v} \sim P[\mathbf{v};\mathcal{G}]$, $\mathbf{u} \sim P[\mathbf{u}\|\mathbf{v};\mathcal{G}]$ $\mathbf{w} \to \mathbf{w} + \epsilon(\mathbf{v}-\mathbf{f}(\mathbf{w}+\mathbf{W}\cdot\mathbf{u}))$ $\mathbf{W} \to \mathbf{W} + \epsilon(\mathbf{v}-\mathbf{f}(\mathbf{w}+\mathbf{W}\cdot\mathbf{u}))\mathbf{u}$

Table 1: All models are discussed in detail in the text, and the forms quoted are just for the simplest cases. $\mathcal{N}(\mathbf{u};\mathbf{g},\mathbf{\Sigma})$ is a multivariate Gaussian distribution with mean $\mathbf{g}$ and covariance matrix $\mathbf{\Sigma}$ (for $\mathcal{N}(\mathbf{u};\mathbf{g},\Sigma)$, the variance of each component is Σ). For the sparse coding network, σ^2 is a target for the variances of each output unit. For the Helmholtz machine, $f(c) = 1/(1+\exp(-c))$, and the symbol $\sim$ indicates that the indicated variable is drawn from the indicated distribution. Other symbols and distributions are defined in the text.

10.7 Annotated Bibliography

The literature on unsupervised representational learning models is extensive. Recent reviews, from which we have borrowed include **Hinton (1989)**, **Bishop (1995)**, **Hinton & Ghahramani (1997)**, and **Becker & Plumbley (1996)**. These references also describe unsupervised learning methods such as IMAX (Becker & Hinton, 1992) that find statistical structure in the inputs directly rather than through causal models (see also projection pursuit, Huber, 1985). The field of belief networks or graphical statistical models (**Pearl, 1988**; Lauritzen, 1996; **Jordan, 1998**) provides an even more general framework for probabilistic generative models. Apart from **Barlow** (1961, **1989**), early inspiration for unsupervised learning models came from Uttley (1979) and Marr (1970), and from the adaptive resonance theory (ART) of Carpenter & Grossberg (1991).

Analysis by synthesis (e.g., Neisser, 1967), to which generative and recognition models are closely related, was developed in a statistical context by Grenander (1995), and was suggested by Mumford (1994) as a way of understanding hierarchical neural processing. Suggestions made in MacKay (1956), Pece (1992), Kawato et al. (1993), and Rao & Ballard (1997) can be seen in a similar light.

Nowlan (1991) introduced the mixtures of Gaussians architecture into neural networks. Mixture models are commonplace in statistics and are described by Titterington et al. (1985).

Factor analysis is described by Everitt (1984). Some of the differences and similarities between factor analysis and principal components analysis are brought out in Jolliffe (1986), Tipping & Bishop (1999), and Roweis & Ghahramani (1999). Rubin & Thayer (1982) discusses the use of EM for factor analysis. Roweis (1998) presents EM for principal components analysis.

Neal & Hinton (1998) describes $\mathcal{F}$ and its role in the EM algorithm (Baum et al., 1970; Dempster et al., 1977). EM is closely related to mean field methods in physics, as discussed in Jordan et al. (1998) and Saul & Jordan (2000). Hinton & Zemel (1994) and Zemel (1994) use $\mathcal{F}$ for unsupervised learning in a backpropagation network called the autoencoder, and these results are related to minimum description length coding (Risannen, 1989). Hinton et al. (1995) and Dayan et al. (1995) use $\mathcal{F}$ in the Helmholtz machine and the associated wake-sleep algorithm.

Olshausen & Field (1996) presents the sparse coding network based on Field's (1994) general analysis of sparse representations, and Olshausen (1996) develops some of the links to density estimation. Independent components analysis (ICA) was introduced as a problem by Herrault & Jutten (1986). The version of the ICA algorithm that we described is due to Bell & Sejnowski (1995) and Roth & Baram (1996), using the natural gradient trick of Amari (1999). The derivation we used is from MacKay (1996). Pearlmutter & Parra (1996) and Olshausen (1996) also derive maximum likelihood

interpretations of ICA. Multi-resolution decompositions were introduced into computer vision in Witkin (1983) and Burt & Adelson (1983). Wavelet analysis is reviewed in Daubechies (1992), **Simoncelli et al. (1992)**, and **Mallat (1998)**.