Chapter 11

Representational Learning

11.1 Introduction

A vast bulk of computations in the brain is devoted to the task of taking information that is represented in the activities of groups of units, and *re-representing* it in different ways. The new representations can either be in the activities of other units, or in the activities of the same units after some interactions in a network. We have already seen many examples of this – for instance, in the way that the activities of photoreceptors are represented and re-represented by the activities of cells from retinal ganglion cells to striate cortical cells and beyond. We have also already seen some of the empirical properties of neural representations that are created – particularly in terms of population codes, *ie* the way that multiple cells jointly conspire to represent quantities of interest. This chapter studies neural representations from a computational perspective, asking what goals are served by particular representations and how appropriate representations can be learned.

Many suggestions have been made as to what makes representations good – suggestions that collectively overlap and conflict. Most start from the premise that there is regular *structure* underlying the noisy and extremely high dimensional sensory input, and that good representational schemes extract just this structure and eliminate noise and anything else extraneous. The suggestions differ in their (usually implicit) expectations about what forms this structure takes and how it can be revealed. It is clear that prior knowledge about particular input distributions (*eg* information about the difference between visual and auditory input) is likely to be critical. Nevertheless, there have been many suggestions for somewhat abstract measures of representational quality, and it is on these that we focus. Much of the inspiration underlying the resulting measures is due to Barlow (see 1989), who has made a number of influential suggestions, many of which have been adapted and formalised.

One rather abstract and general notion comes from considering *casual models.* The idea is that an input, such as the image of a scene, has distal *causes,* such as objects at given locations illuminated by some particular lighting scheme observed from a particular viewing direction. These causes make for the structure in the input, and, since inferences and decisions are normally based on underlying causes, make for appropriate representations of input. To put it another way, the images that we see live in an $O(10^8)$-dimensional space which has one dimension for each photoreceptor (plus one dimension for time). However, the set of all images we might ever naturally see is much smaller, since it is constrained by how images are actually generated. The image generation process specifies the natural coordinates for the observable images, in terms of the things that cause them, and it is these coordinates that we seek to use in order to represent new images.

More concrete notions about extracting the structure in the input (and therefore also about extracting causes) include finding *lower dimensional* projections of the input that nevertheless convey *most of the information* in the input, or *code* the input in a way that makes it cheap to communicate, finding projections whose activities are *mutually independent* (also known as *factorial*) or *sparse,* or with distributions that are *statistically unlikely* and therefore potentially unusually informative. A major source of statistical structure in the input comes from groups of transformations – for instance translation and scale for visual images. Extracting the structure correctly requires respecting (or discovering) these transformations.

We consider a number of learning methods that are classed as being *unsupervised* or *self-supervised*, in the sense that no information other than the patterns themselves is explicitly provided to guide the representational choice. This is by contrast with supervised methods described in chapter 8, which learn appropriate representations in the service of particular tasks. A main reason for studying unsupervised methods is that they are likely to be much more common than supervised ones. For instance, there is only a derisory amount of supervisory information available to learn how to perform a task such as object recognition that is based on the activities in the $O(10^8)$-dimensional space of activities of the photoreceptors. Unsupervised learning is also often used to try to capture what is happening during activity-dependent development. We saw in chapters 8 and 9 mathematical models that capture the nature and plasticity of the development of cells in visual cortex on the basis of the activity of input neurons. We seek to provide quantifiable goals for this unsupervised process. As will become evident, in neural terms, unsupervised learning methods are still in their infancy – they are only barely capable of generating the forms of sophisticated population code representations that we have discussed and dissected, and even central concepts, such as the notion of causes in causal models and the groups of relevant transformations that particular inputs undergo are only relatively weakly formulated.

The two things that unsupervised learning methods have to work with are observed input patterns or cases $\mathbf{u}$ (we typically assume that either $\mathbf{u} \in \Re^{n_u}$ or $\mathbf{u} \in \{0,1\}^{n_u}$, where n_u is the dimensionality of the input) and some explicit or implicit *a priori* information as to what representational learning should achieve. In general we assume that the inputs are generated as independent random draws from an underlying probability distribution $P_I[\mathbf{u}]$, and seek to learn from a set of such inputs an appropriate way of representing future cases $\mathbf{u}$ drawn from the same distribution. We assume that there is an ample supply of input patterns, and so use the collection of actual input examples and the full probability distribution $P_I[\mathbf{u}]$ interchangeably.

Most of the representational schemes that are based on causal models can be learnt through *density estimation, ie* by fitting to the probability distribution $P_I[\mathbf{u}]$ of the inputs, a probabilistic model in which the causes play a crucial role. The criteria that causes should satisfy are generally directly implemented in the *architecture* of the probabilistic causal model. This is an essential point – fitting an unconstrained probabilistic model (for instance a look-up table) to the inputs might be an excellent way of modeling the distribution, but it has nothing to say about causes. Further, it turns out to be possible to reinterpret many representational learning schemes that were not originally suggested in terms of causal models and density estimation as if they had been, generally by making different assumptions about the architecture of the causal model.

However, there are representational learning schemes which cannot be treated as arising from fitting a causal model, for the essential reason that learning full causal models is far too stringent a requirement for representations. For instance, we are generally quite uninterested in the portions of images caused by texture patterns, such as the design of a carpet, or the seams in marble. However, these can clearly be visual inputs too, and so if we were trying to build a density estimator that captures the whole of the input, then we would have to devote effort to learning these otherwise uninteresting causes. Some of the consequences of this problem will become apparent in discussing multi-resolution models of visual input. Alternative representational learning schemes work by searching the input directly for statistically interesting or relevant features rather than doing so indirectly using a causal model. One example is projection pursuit. This is based on the fact (essentially due to the central limit theorem) that most low dimensional projections of high dimensional inputs have approximately Gaussian statistics. Therefore projections which have non-Gaussian statistics are likely to report something specially significant or interesting about the inputs. Projection pursuit defines the goal of representational learning in terms of finding projections with non-Gaussian statistics. It turns out the the sliding threshold Hebbian rule described in equations 8.34 and 8.35 of chapter 8 can be seen as implementing a particular form of projection pursuit. Another example is a learning rule called IMAX. This seeks to maximise the mutual information between the representations of disparate parts of the input, on the grounds that the information *common* between separate parts of the input is exactly the information that should be made explicit in a representation of the input. We do not discuss such models of representational learning further.

11.2 Density Estimation

Many of the density estimation architectures, or approximations to them, can be understood in terms of figure 11.1. This shows a two layer network, with inputs $\mathbf{u}$ and representation units $\mathbf{v}$, which are intended to represent the causes of the inputs $\mathbf{u}$. There are two sets of parameters: $\mathcal{G}$ and $\mathcal{W}$, although some models use only one of the two. Parameters $\mathcal{G}$ control the causal model, which is also called a *generative* or *synthetic* model. Parameters $\mathbf{W}$ control the inverse to the causal model, which is also called a *recognition* or *analytical* model.

Models differ as to whether and how the units are actually *activated* on the basis of these parameters – for some, the parameters should not be treated as simple weights in a uni-directional or bidirectional network. Some models also allow more than two layers or have connections between the units within a single layer.

The simplest models use only parameters $\mathbf{W}$ as the weights in a completely standard unidirectional neural net. More complicated models use only $\mathbf{G}$, but, because $\mathbf{G}$ are the parameters for a top-down neural network, they require a different mechanism for setting the activations of units $\mathbf{v}$ in response to an input $\mathbf{u}$. The most complicated models we consider employ a combination of parameters $\mathbf{G}$ and $\mathbf{W}$, but at any particular time the activations of the units are controlled by *either* $\mathbf{G}$ *or* $\mathbf{W}$, but not both.

Although it is natural to identify top-down connections in the cortex (*eg* from area V2 to V1) with the generative model and bottom-up connections in cortex (*eg* from V1 to V2) with the recognition model, and to view top-down effects in perception (such as biases that come from expecting to see a particular object or hear a particular word) as being mediated by the top-down weights, the models are rather too abstract to speak to this.

The generative model indicates what the input $\mathbf{u}$ would be like (at least probabilistically) for a given set of causes $\mathbf{v}$. A general parameterised form of generative model is the probability distribution

$$P[\mathbf{u}|\mathbf{v};\mathcal{G}] \tag{11.1}$$

where parameters $\mathcal{G}$ control the way that causes $\mathbf{v}$ can determine images $\mathbf{u}$ (for example, what the image of an orange might look like to a photoreceptor). To be able to capture the complete range of observed inputs $\mathbf{u}$, the generative model must also specify what the probabilities are of the causes (what the probability is that there is an orange in the scene). This is also a parameterised probability distribution

$$P[\mathbf{v};\mathcal{G}] \tag{11.2}$$

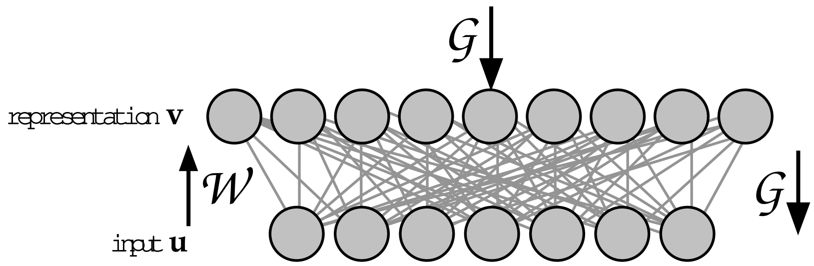

Figure 11.1: Simple generic architecture for maximum likelihood density estimation. Parameters $\mathcal{G}$ (which need not necessarily be synaptic weights between units) govern the joint distribution $P[\mathbf{u}, \mathbf{v}; \mathcal{G}]$ over the representation units $\mathbf{v}$ and the input units $\mathbf{u}$. The task is to make $\sum_{\mathbf{v}} P[\mathbf{u}, \mathbf{v}; \mathcal{G}]$ close to the empirical $P_I[\mathbf{u}]$. Many architectures (though not the Boltzmann machine) have a top-down structure with a natural factorisation $P[\mathbf{u}, \mathbf{v}; \mathcal{G}] = P[\mathbf{v}; \mathcal{G}]P[\mathbf{u}|\mathbf{v}; \mathcal{G}]$. For some architectures, bottom-up parameters $\mathcal{W}$ help determine the approximation $Q[\mathbf{v}; \mathbf{u}]$ to the statistical inverse $P[\mathbf{v}|\mathbf{u}; \mathcal{G}]$ of the generative model. See text for more details.

which is sometimes called the *prior* distribution over causes. Since the causes $\mathbf{v}$ for particular inputs $\mathbf{u}$ are unknown, they are sometimes called *hidden* or *latent* variables.

The probabilistic generative model in equations 11.1 and 11.2 accords a total (also called a *marginal*) probability to a particular input $\mathbf{u}$ of:

$$P[\mathbf{u}; \mathcal{G}] = \sum_{\mathbf{v}} P[\mathbf{u}, \mathbf{v}; \mathcal{G}] = \sum_{\mathbf{v}} P[\mathbf{v}; \mathcal{G}]P[\mathbf{u}|\mathbf{v}; \mathcal{G}] \,, \tag{11.3}$$

where the sum is over all possible values of $\mathbf{v}$, and is really an integral if $\mathbf{v}$ live in a continuous space.

In the same way that a standard neural net has (i) a connectivity structure (and activation functions) and (ii) a set of variable weights, probabilistic generative models usually have two components – an overall *architecture* and variable *parameters* within that architecture. In this chapter, we study architectures that parameterise the distribution in equation 11.3 in very different ways, and then seek the best values of the parameters within each architecture for a given set of inputs. A huge variety of different architectures has been suggested, only some of which even fit into the simple two-layer scheme in figure 11.1 – we only consider some of the key varieties.

If the generative model in equations 11.1 and 11.2 correctly captures the causes which underlie the production of an input $\mathbf{u}$, then $P[\mathbf{u}; \mathcal{G}]$ should exactly equal $P_I[\mathbf{u}]$. We would then have established how causes, of the variety that the architecture entertains, *could* generate the observed input. It is therefore appropriate to learn parameters $\mathcal{G}$ by minimising the discrepancy between the distributions $P[\mathbf{u}; \mathcal{G}]$ and $P_I[\mathbf{u}]$. A natural measure of the discrepancy between $P_I[\mathbf{u}]$ and $P[\mathbf{u}; \mathcal{G}]$ is the Kullback-Leibler divergence:

$$KL\left[P_I[\mathbf{u}],\, P[\mathbf{u};\,\mathcal{G}]\right] = \sum_{\mathbf{u}} P_I[\mathbf{u}] \log \frac{P_I[\mathbf{u}]}{P[\mathbf{u};\,\mathcal{G}]} \tag{11.4}$$

that we discussed in chapter 4, where the sum is over all **u** (remembering we assume that the training set is exhaustive). Assuming that the cases are generated independently, we have that the log likelihood of the training data under the generative model averaged over the training set is

$$\begin{aligned}
\langle \log P[\mathbf{u};\,\mathcal{G}]\rangle_{P_I[\mathbf{u}]} &= \sum_{\mathbf{u}} P_I[\mathbf{u}] \log P[\mathbf{u};\,\mathcal{G}] && (11.5)\\
&= \sum_{\mathbf{u}} P_I[\mathbf{u}] \log \frac{P[\mathbf{u};\,\mathcal{G}]}{P_I[\mathbf{u}]} - \mathcal{H}\left[P_I[\mathbf{u}]\right] && (11.6)\\
&= -KL\left[P_I[\mathbf{u}],\, P[\mathbf{u};\,\mathcal{G}]\right] - \mathcal{H}\left[P_I[\mathbf{u}]\right] && (11.7)
\end{aligned}$$

where $\mathcal{H}[P_I[\mathbf{u}]]$ is the entropy of the training set distribution and is independent of $\mathcal{G}$. This means that *minimising* the Kullback-Leibler divergence is just the same as maximising the average log likelihood of the observed data under the model. Choosing $\mathcal{G}$ to minimise the Kullback-Leibler divergence is therefore called a maximum likelihood (ML) method. In general, the likelihood will have many local maxima in the parameters, and most of the methods can only be guaranteed to find a local rather than a global maximum.

Given appropriate values of the parameters learned on the basis of a training set, the key remaining task is to generate a representation for a new input **u** in terms of the causes **v**. Given the probabilistic nature of the model, there will in general be a whole distribution over the causes:

$$P[\mathbf{v}|\mathbf{u};\,\mathcal{G}] = \frac{P[\mathbf{u},\mathbf{v};\,\mathcal{G}]}{\sum_{\mathbf{v}'} P[\mathbf{u},\mathbf{v}';\,\mathcal{G}]} \tag{11.8}$$

which is called the posterior distribution over **v** given **u**. If it is required to extract a single value **v** that represents **u**, this can be done using any one of a set of possible loss functions which report the cost of choosing one particular representation when a different one is actually true (loss functions for statistical inference are discussed in chapter 3). By contrast with the generative or synthetic model $P[\mathbf{u},\mathbf{v};\,\mathcal{G}]$, the posterior distribution in equation 11.8 is also called the analytical or recognition model, since it specifies how to analyse an input **u** in terms of its possible causes or to recognise what the input contains. As is evident in equation 11.8, the analytical model is a form of statistical inverse of the synthetic model, and the pair of models together performs the technique of analysis by synthesis. The recognition distribution is not only important for novel inputs. Many methods for learning optimal parameters $\mathcal{G}^*$ for the generative model require one to be able to calculate $P[\mathbf{v}|\mathbf{u};\,\mathcal{G}]$ for the patterns **u** in the training set, and for arbitrary $\mathcal{G}$.

It turns out that for many architectures of interest, the recognition distribution is very difficult to calculate exactly. The trouble stems from the fact that there are so many possible interacting causes for most inputs, that it is computationally crippling to calculate the sum in the denominator of equation 11.8 exactly. One option (adopted by the Boltzmann machine) is to try to base inference and learning on *samples* $\mathbf{v}$ extracted from the distribution $P[\mathbf{v}|\mathbf{u};\mathcal{G}]$. In some cases, one can expect to be able to get samples that are quite faithful to the distribution with little computational effort even if calculating the full distribution is much too expensive. The other option is to try to approximate the distribution $P[\mathbf{v}|\mathbf{u};\mathcal{G}]$ with an alternative distribution $\mathcal{Q}[\mathbf{v};\mathbf{u}]$, which can be chosen to be tractable to calculate and use, and can be optimised to make it as close as possible to $P[\mathbf{v}|\mathbf{u};\mathcal{G}]$. This alternative distribution can be used to provide the representation of new inputs, and it can also be used to learn the parameters $\mathcal{G}^*$ that approximately minimise the Kullback-Leibler divergence in equation 11.4 and approximately maximise the likelihood in equation 11.5 in the first place.

Consider a single input case $\mathbf{u}$. By Jensen's inequality (appendix ?, chapter 4),

$$\log P[\mathbf{u};\mathcal{G}] = \log \sum_{\mathbf{v}} P[\mathbf{u},\mathbf{v};\mathcal{G}] \geq \sum_{\mathbf{v}} \mathcal{Q}[\mathbf{v};\mathbf{u}] \log \frac{P[\mathbf{u},\mathbf{v};\mathcal{G}]}{\mathcal{Q}[\mathbf{v};\mathbf{u}]} \tag{11.9}$$

with equality if and only if $\mathcal{Q}[\mathbf{v};\mathbf{u}] \propto P[\mathbf{u},\mathbf{v};\mathcal{G}]$. But since $\sum_{\mathbf{v}'} \mathcal{Q}[\mathbf{v}';\mathbf{u}] = 1$, this equality condition makes

$$\mathcal{Q}[\mathbf{v};\mathbf{u}] = \frac{P[\mathbf{u},\mathbf{v};\mathcal{G}]}{\sum_{\mathbf{v}'} P[\mathbf{u},\mathbf{v}';\mathcal{G}]} = P[\mathbf{v}|\mathbf{u};\mathcal{G}]. \tag{11.10}$$

which is exactly the recognition distribution. Write:

$$\mathcal{F}[\mathbf{u},\mathcal{Q};\mathcal{G}] = -\sum_{\mathbf{v}} \mathcal{Q}[\mathbf{v};\mathbf{u}] \log P[\mathbf{u},\mathbf{v};\mathcal{G}] + \sum_{\mathbf{v}} \mathcal{Q}[\mathbf{v};\mathbf{u}] \log \mathcal{Q}[\mathbf{v};\mathbf{u}] \tag{11.11}$$

which is just minus the right hand side of equation 11.9 and so is an *upper* bound on the negative log likelihood for case $\mathbf{u}$. One standard maximum likelihood technique called expectation-maximisation (EM; Dempster, Laird & Rubin, 1977) consists of a form of coordinate-wise descent of

$$\mathcal{F}[\{\mathcal{Q}\};\mathcal{G}] = \langle \mathcal{F}[\mathbf{u},\mathcal{Q};\mathcal{G}] \rangle_{P_I[\mathbf{u}]} \tag{11.12}$$

The expectation step consists of maximising $-\mathcal{F}[\mathbf{u},\mathcal{Q};\mathcal{G}]$ with respect to $\mathcal{Q}$ separately for every $\mathbf{u}$. The maximisation step consists of maximising $-\mathcal{F}[\{\mathcal{Q}\};\mathcal{G}]$ with respect to $\mathcal{G}$, keeping these $\mathcal{Q}$ constant. If maximisation with respect to each $\mathcal{Q}$ in the expectation step is complete, then this determines the true recognition distribution $P[\mathbf{v}|\mathbf{u};\mathcal{G}]$ for each $\mathbf{u}$. Furthermore, because of the coordinate-wise nature of the optimisation, the true log likelihood of the data increases over each complete expectation-maximisation step. For most generative models, it is intractable to make optimisation with respect to $\mathcal{G}$ complete during the maximisation step, but the monotonic improvement in the likelihood remains true. If $\mathcal{Q}$ is constrained during the expectation step and so may fall short of $P[\mathbf{v}|\mathbf{u}; \mathcal{G}]$, then only a bound on the log likelihood of the data is increased over each complete step (Neal & Hinton, 1998). There is almost always a tradeoff between the flexibility and the computational cost of $\mathcal{Q}[\mathbf{v}; \mathbf{u}]$.

It is typical to use an on-line method for optimisation, choosing a case $\mathbf{u}$ either systematically or at random, minimising $\mathcal{F}[\mathbf{u}, \mathcal{Q}; \mathcal{G}]$ with respect to $\mathcal{Q}[\mathbf{v}; \mathbf{u}]$, giving distribution $\mathcal{Q}^*$, and then calculating $\nabla_{\mathcal{G}}\mathcal{F}[\mathbf{u}, \mathcal{Q}^*; \mathcal{G}]$ for the optimal recognition distribution $\mathcal{Q}^*$. Parameters $\mathcal{G}$ are then changed in the light of this derivative, performing a form of stochastic gradient descent:

$$\Delta\mathcal{G} \propto -\nabla_{\mathcal{G}}\mathcal{F}[\mathbf{u}, \mathcal{Q}; \mathcal{G}] \,, \tag{11.13}$$

although more powerful methods could also be used. In terms of figure 11.1, some distributions $\mathcal{Q}$ use a set of parameters $\mathcal{W}$ directly, which are learned across input cases. Other distributions $\mathcal{Q}$ are determined by different procedures that do not use a separate set of parameters.

Given this general scheme, what remains is to specify particular network structures and priors over $\mathbf{v}$, and to show how different learning rules can solve the two tasks presented of determining the optimal parameters $\mathcal{G}^*$ and the optimal distribution $\mathcal{Q}^*$ over $\mathbf{v}$ given $\mathbf{u}$ and particular $\mathcal{G}$. In order to cover a broad spectrum of models, we study in this light factor analysis, principal components analysis, mixture-of-Gaussians clustering, sparse coding generative models, maximum likelihood independent components analysis, the Boltzmann machine, and the Helmholtz machine. These are all instances of directed or undirected *belief networks,* a class of statistical models that is presently the focus of substantial theoretical and practical investigation. Appendix A summarises the various generative and recognition models.

11.3 Factor Analysis

So far, we have specified density estimation in complete generality. To motivate the discussion of particular architectures and their learning rules, we consider the special case of *factor analysis.* Factor analysis is a conventional statistical model in which the correlations among a set of inputs (such as scores of students in tests of a variety of subjects) are explained by a small number of independent hidden factors (such as different supposed components or quotients of intelligence). In our terms, the intelligence quotients are the underlying *causes* of the inputs, and we seek the influence of each aspect of intelligence on the result of each test (*ie* $\mathcal{G}$) and, given a particular set of scores $\mathbf{u}$, the distribution over the underlying intelligence quotients $\mathbf{v}$. Of course, the application to intelligence quotients is controversial – but the statistical model is sound.

In particular, we consider the case of linear, Gaussian factor analysis, for which the probabilities in equations 11.1 and 11.2 are all Gaussian, given by:

$$\begin{aligned} P[\mathbf{v};\mathcal{G}] &\sim \mathcal{N}[\mathbf{0},\sigma^2\mathbb{I}] \\ P[\mathbf{u}|\mathbf{v};\mathcal{G}] &\sim \mathcal{N}[\mathbf{G}\mathbf{v},\Psi] \end{aligned} \tag{11.14}$$

where $\mathbb{I}$ is the identity matrix (of an appropriate dimension), matrix $\Psi = \text{diag}\left(\tau_1^2, \tau_2^2, \ldots, \tau_n^2\right)$, $\mathcal{G} = \{\mathbf{G}, \sigma^2, \Psi\}$ is the collection of all the generative parameters and $\mathcal{N}[\mathbf{a}, \Omega]$ is a multivariate Gaussian distribution function with mean $\mathbf{a}$, covariance matrix Ω and the functional form, for example:

$$P[\mathbf{u}|\mathbf{v};\mathcal{G}] \propto \frac{1}{\sqrt{|\Psi|}} e^{-(\mathbf{u}-\mathbf{G}\mathbf{v})^T\Psi^{-1}(\mathbf{u}-\mathbf{G}\mathbf{v})/2} \tag{11.15}$$

This prior and structure specifies that the individual components v_a of $\mathbf{v}$ are factorial or mutually independent (since they are Gaussian and uncorrelated). Furthermore, since Ψ is diagonal, it specifies that any correlations between the components of $\mathbf{u}$ should be explained by the causes $\mathbf{v}$, through the mean value $\mathbf{G}\mathbf{v}$ of the generative model for $\mathbf{u}$. The parameters of the model are in general only well determined if $\mathbf{v}$ has fewer dimensions than $\mathbf{u}$. To put it in terms of the desiderata for causes listed above, factor analysis seeks causes that should be independent and low dimensional, albeit, in this simple case, also having a particular linear relationship with the inputs.

We assume that distribution $P_I[\mathbf{u}]$ has zero mean (non-zero means can simply be accommodated using appropriate biases). By direct calculation, equation 11.14 makes the unconditional distribution for $\mathbf{u}$

$$P[\mathbf{u};\mathcal{G}] \sim \mathcal{N}\left[\mathbf{0}, \sigma^2\mathbf{G}\mathbf{G}^T + \Psi\right] \tag{11.16}$$

and so the task for factor analysis is to match the covariance matrix of the inputs $\mathbf{C} = \langle \mathbf{u}\mathbf{u}^T \rangle$ (averaging over $P_I[\mathbf{u}]$) with $\sigma^2\mathbf{G}\mathbf{G}^T + \Psi$. This illustrates the fact that linear factor analysis only takes account of second order statistical structure in the data. We argue later that in general, this is insufficient.

Since $\mathbf{u}$ and $\mathbf{v}$ are jointly Gaussian, the posterior or recognition distribution $P[\mathbf{v}|\mathbf{u};\mathcal{G}]$ is also Gaussian $\mathcal{N}[\mathbf{W}^*\mathbf{u}, \Sigma^*]$. It is straightforward to show that:

$$\mathbf{W}^* = \left(\frac{\mathbb{I}}{\sigma^2} + \mathbf{G}^T\Psi^{-1}\mathbf{G}\right)^{-1} \mathbf{G}^T\Psi^{-1} \tag{11.17}$$

$$\Sigma^* = \left(\frac{\mathbb{I}}{\sigma^2} + \mathbf{G}^T\Psi^{-1}\mathbf{G}\right)^{-1} \tag{11.18}$$

Σ^* does not depend on the inputs $\mathbf{u}$ and so bottom-up weights $\mathbf{W}^*$ as in figure 11.1 can be used to specify the mean of the recognition distribution.

Equations 11.17 and 11.18, as the recognition distribution, are the output of the expectation step of the EM algorithm. Being completely explicit, the full maximisation step requires finding weights $\mathcal{G}$ that minimise

$$-\sum_{\mathbf{u}} P_I[\mathbf{u}] \sum_{\mathbf{v}} P[\mathbf{v}|\mathbf{u}; \mathcal{G}] \log P[\mathbf{u}, \mathbf{v}; \mathcal{G}] \,. \tag{11.19}$$

Further,

$$\begin{aligned} -\log P[\mathbf{u}, \mathbf{v}; \mathcal{G}] \;\; = \;\; & \tfrac{1}{2\sigma^2}|\mathbf{v}|^2 + \tfrac{1}{2}(\mathbf{u} - \mathbf{G}\mathbf{v})^T \Psi^{-1}(\mathbf{u} - \mathbf{G}\mathbf{v}) + \\ & \tfrac{1}{2}\left(\log \sigma^2 + \textstyle\sum_{i=1}^{n} \log \tau_i^2\right) + \mathcal{K} \end{aligned} \tag{11.20}$$

where $\mathcal{K}$ is constant with respect to $\mathcal{G}$. Therefore, solving the resulting unconstrained quadratic optimisation problem with respect to $\mathcal{G}$ gives as the maximisation step:

$$\tau_a^2 \;\; \rightarrow \;\; (\mathbb{I} - \mathbf{G}\mathbf{W}^*)\mathbf{C}(\mathbb{I} - \mathbf{G}\mathbf{W}^*)^T + \mathbf{G}\Sigma^*\mathbf{G}^T \tag{11.21}$$

$$\mathbf{G} \;\; \rightarrow \;\; \mathbf{C}\mathbf{W}^{*T}(\mathbf{W}^*\mathbf{C}\mathbf{W}^{*T} + \Sigma^*)^{-1} \tag{11.22}$$

We see later a variety of non-linear extensions and approximations to factor analysis in which E and / or M steps of EM are performed only partially and without requiring explicit matrix inversion.

The expression in equation 11.20 is revealing. The first term comes from the prior distribution over $\mathbf{v}$. The second term is called the (weighted) reconstruction error between the actual input $\mathbf{u}$ and the mean of the input that would be expected on the basis of representation $\mathbf{v}$. The appropriate parameters $\mathcal{G}$ are those that allow low reconstruction error using representations that are likely under the prior distribution over $\mathbf{v}$ (which is given in equation 11.14).

11.4 Principal Components Analysis

Consider a simple case of factor analysis in which there is only one cause or factor (*ie* $\mathbf{v}$ is one dimensional) and also in which the residual variances of all the input components are forced to be the same, *ie* $\Psi = \tau^2\mathbb{I}$. Let the generative weights be the *row* vector $\mathbf{g}$ and, exclusively for the case of a single output unit v, the recognition weights be the row vector $\mathbf{w}$. Equations 11.17 and 11.18 imply that

$$\mathbf{w}^* = \frac{\sigma^2}{\tau^2 + \sigma^2\|\mathbf{g}\|^2}\mathbf{g} \quad \text{and} \quad \Sigma^* = \frac{\sigma^2\tau^2}{\tau^2 + \sigma^2\|\mathbf{g}\|^2} \tag{11.23}$$

Since these give the true recognition distribution of v given a particular $\mathbf{u}$, evaluating the free energy in equation 11.11 also gives the true negative log probability of $\mathbf{u}$ given the generative model parameters $\mathcal{G}$. Averaging this free energy over $P_I[\mathbf{u}]$ and retaining only the terms in $\mathcal{G}$ leads to:

$$-\frac{\sigma^2}{2\tau^2(\tau^2 + \sigma^2\|\mathbf{g}\|^2)}\mathbf{g}^T\mathbf{C}\mathbf{g} + \frac{1}{2}\left(\log\sigma^2 + n\log\tau^2 + \log\frac{\sigma^2\tau^2}{\tau^2 + \sigma^2\|\mathbf{g}\|^2}\right) \tag{11.24}$$

where $\mathbf{C}$ is the covariance matrix of the inputs, assuming, again, that u_b have zero mean.

The expression in equation 11.24 has a very simple form since the only term that depends on the direction of $\mathbf{g}$ rather than its magnitude is $-\mathbf{g}^T\mathbf{C}\mathbf{g}$. We perform maximum likelihood density estimation by minimising this with respect to $\mathbf{g}$. We focus on finding the optimal direction – finding the optimal magnitude is just messier.

We study two versions of exactly this optimisation problem in chapter 8. The conclusion is that the optimal generative weights (and therefore, by equation 11.23, the optimal recognition weights) point in the direction of the principal component of the covariance matrix of the data. Therefore, we have derived a simple case of the statistical technique of principal components analysis (PCA) from a simple case of factor analysis. This version of PCA suggests projecting individual $\mathbf{u}$ onto the principal eigenvector of the covariance matrix of all the data.

In general, if $\Psi = \tau^2\mathbb{I}$ in the standard factor analysis model of equation 11.14, then the optimal *generative* weights span the space of as many principal components as there are causes or factors. However, since equation 11.17 shows that the optimal recognition weights are not just scaled versions of the generative weights, the representations $P[\mathbf{v}|\mathbf{u};\mathcal{G}]$ are not straightforwardly related to the projection of the input $\mathbf{u}$ onto the principal component subspace of the data unless there is only one factor. Nevertheless, the information about $\mathbf{u}$ retained in the mean value of $\mathbf{v} = \mathbf{W}^*\mathbf{u}$ in this factor analysis solution is the same as the information retained by PCA in the sense that one can calculate from this mean value the answer that would be given by PCA. In many cases, the first few principal components carry most of the interesting information about the data, and therefore, putatively, form a good representation of it.

One can also view true PCA as an approximate form of density estimation. Imagine representing a point $\mathbf{u}$ directly by a single deterministic linear projection $v = v(\mathbf{u}) = \mathbf{w}\cdot\mathbf{u}$. Using a deterministic representation is equivalent to choosing a delta function as the approximate posterior:

$$Q[v;\mathbf{u},\mathbf{w}] = \delta(v - \mathbf{w}\cdot\mathbf{u}) \tag{11.25}$$

or forcing $\Sigma \to 0$. Consider adopting the model in expression 11.14 but with $\sigma^2 \gg \tau^2$. This makes the prior for v relatively unimportant. Then, from equations 11.11 and 11.20, approximate density estimation requires minimising:

$$-\langle \log P[\mathbf{u}, v; \mathcal{G}]\rangle_{P_I[\mathbf{u}]} = \mathcal{K} + \frac{1}{2\tau^2}\left\langle |\mathbf{u} - \mathbf{g}v|^2 \right\rangle_{P_I[\mathbf{u}]} \tag{11.26}$$

where $\mathcal{K}$ is a constant. Minimising this exactly requires minimising reconstruction error with a linear reconstruction. We saw in chapter 8 that this is one of the properties of PCA. In the case of multiple components, the optimal representation $\mathbf{v}$ of an input $\mathbf{u}$ is exactly the projection of $\mathbf{u}$ onto the space spanned by the appropriate number of principal components of the data

Like factor analysis, PCA is based only on first and second order statistics of the input. This is an extremely severe restriction. If the input is whitened, for instance, by retinal processing in some regimes of signal to noise ratio (see chapter 4), the covariance matrix $\mathbf{C}$ may be the identity $\mathbb{I}$. In this case, *neither* PCA nor FA would be able to extract any interesting structure. Many of the architectures discussed below are, by contrast, sensitive to higher order statistical structure.

In chapter 8, we derived two computational properties of the representation of inputs that PCA provides, namely that it minimises reconstruction error given a linear reconstruction, and it maximises the mutual information between the input and output under this linear and Gaussian model. Both of these properties have been suggested as goals for representational learning independently of their connection to PCA, and we have now seen how that they are related to density estimation. Note, however, these properties are not necessarily appropriate by themselves. For instance, it is troubling that the principal component conveys the most information about $\mathbf{u}$ by maximising the variance of v. It is not at all clear that choosing a representation that maximises the variance of the output is appropriate, unless we know that all this variance is signal rather than noise. If the independent input noise variances τ_i^2 are allowed to be different for different inputs, factor analysis is not sensitive to variance so much as covariance between its inputs, so noise will only be reported in v if it is shared among several components of $\mathbf{u}$.

Factor analysis and principal components analysis are two important examples of exact and approximate ML density estimation. We now discuss other exact and approximate methods.

11.5 Clustering

A standard heuristic for good representations is sparsity – *ie* the components of $\mathbf{v}$ should generally be inactive, or at least weakly active. The empirical activity distributions of cortical cells are highly non-Gaussian. Figure 11.2 shows an example of the spike count distribution of a macaque IT cell recorded whilst the monkey was watching relatively unstructured videos. The distribution is close to being *exponential.* Crudely, compared with an equivalent (rectified) Gaussian, this means the cell is more likely to have a low activity level, but can also have quite high activities (the log probability falls off linearly rather than quadratically). These occasional high activities can convey substantial information. Cells in other areas, including V1 are also sparse.

Sparseness can be wasteful in the sense that sparse representations may (though not necessarily) require large numbers of units. However, poten-

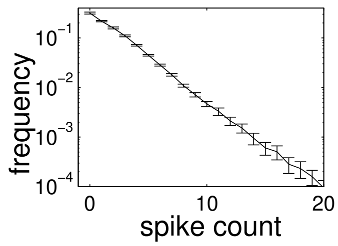

Figure 11.2: Log frequency distribution of the activity of a macaque IT cell in response to a collection of three different video clips. The size of the window used to calculate the spike counts was adjusted separately for each video so that, on average, there were two spikes per window. This normalises differences in the mean activity of the cell. From Baddeley *et al* (1998).

tial downstream processing mechanisms (such as heteroassociative memories) require sparse input activities to perform well, since this minimises interference between different patterns in their synaptic weight matrices. Furthermore, in many cases, there is representational *expansion* rather than the sort of representational *compression* that drives FA and PCA – *eg* there are around 40 cells in primary visual cortex to represent the activity of each cell in the visual thalamus.

FA and PCA are incapable of generating sparse representations – all of the units participate actively in representing each $\mathbf{u}$. An extreme form of sparsity is to allow just a single unit among the $\mathbf{v}$ to participate in the representation of any given input. This amounts to saying that each input $\mathbf{u}$ has exactly one cause (specified by the single active $\mathbf{v}$ unit). This is clearly too stringent at the scale of a whole input (for instance a whole visual scene), but may be more reasonable at smaller scales.

A probabilistic version of this sparseness requirement starts from the usual ML density estimation generative architecture, except that $P[\mathbf{v}]$ is 0 unless just one single $v_a = 1$ and all the others are 0. Define

$$P[v_a = 1] = \pi_a \tag{11.27}$$

where $\pi_a \geq 0$ and $\sum_{a=1}^{n_v} \pi_a = 1$ where there are n_v possible causes. Parameters π_a are called *mixing proportions.* A simple way to model the distribution of $\mathbf{u}$ given that $v_a = 1$ is as a Gaussian:

$$P[\mathbf{u}|v_a = 1; \mathcal{G}] \sim \mathcal{N}[\mathbf{g}^a, \mathbb{I}] \tag{11.28}$$

where $\mathbf{g}^a$ are further generative parameters. More complicated (and also adaptive) covariance matrices could also be used.

Since only one of the output units can be active in this case, it is tractable to compute the recognition distribution:

$$P[v_a = 1|\mathbf{u}; \mathcal{G}] = \frac{\pi_a P[\mathbf{u}|v_a = 1; \mathcal{G}]}{\sum_{a'} \pi_{a'} P[\mathbf{u}|v_{a'} = 1; \mathcal{G}]} \tag{11.29}$$

$$= \frac{\pi_a e^{-|\mathbf{u}-\mathbf{g}^a|^2/2}}{\sum_{a'} \pi_{a'} e^{-|\mathbf{u}-\mathbf{g}^{a'}|^2/2}} \tag{11.30}$$

The values $P[v_a = 1|\mathbf{u}; \mathcal{G}]$ that define the recognition distribution are also called responsibilities, since they measure the relative extent to which v_a is the cause of $\mathbf{u}$.

Using the posterior of equation 11.30, $\mathcal{G}$ should be chosen to minimise

$$\mathcal{F}[\mathbf{u}, \mathcal{Q}; \mathcal{G}] = \sum_a P[v_a = 1|\mathbf{u}; \mathcal{G}] \left(\left|\mathbf{u} - \mathbf{g}^a\right|^2 - \log \pi_a \right). \tag{11.31}$$

Averaging over the distribution of inputs as in equation 11.19 this leads to a learning rule for the $\mathbf{g}^a$:

$$\mathbf{g}^a \rightarrow \frac{\langle P[v_a = 1|\mathbf{u}; \mathcal{G}]\mathbf{u}\rangle_{P_I[\mathbf{u}]}}{\langle P[v_a = 1|\mathbf{u}; \mathcal{G}]\rangle_{P_I[\mathbf{u}]}}\,, \tag{11.32}$$

and, imposing the condition that $\sum_a \pi_a = 1$, that

$$\pi_a \rightarrow \langle P[v_a = 1|\mathbf{u}; \mathcal{G}]\rangle_{P_I[\mathbf{u}]} \tag{11.33}$$

Stochastic gradient descent in $\mathcal{F}$ is another possibility. This entails drawing $\mathbf{u}$ at random from $P_I[\mathbf{u}]$ and then changing

$$\mathbf{g}^a \rightarrow \mathbf{g}^a - \epsilon \nabla_{\mathbf{g}^a} \mathcal{F}[\mathbf{u}, \mathcal{Q}; \mathcal{G}] \tag{11.34}$$

$$= \mathbf{g}^a + \epsilon P[v_a = 1|\mathbf{u}; \mathcal{G}](\mathbf{u} - \mathbf{g}^a) \tag{11.35}$$

where ϵ is a learning rate. The responsibilities just weight the amount of learning. The other factor is the reconstruction error in the match between $\mathbf{u}$ and $\mathbf{g}^a$. Equation 11.35 is another form of the delta rule.

It is a little more difficult to adapt the mixing proportions, π_a, since they satisfy two constraints: $\pi_a \geq 0$ and $\sum_a \pi_a = 1$. It is convenient to use an alternative parameterisation

$$\pi_a = \frac{e^{\rho_a}}{\sum_{a'} e^{\rho_{a'}}} \tag{11.36}$$

in which these constraints are automatically satisfied. Gradient descent of $\mathcal{F}$ from equation 11.31 in the new parameters ρ_a requires

$$\rho_a \rightarrow \rho_a + \epsilon(P[v_a = 1|\mathbf{u}; \mathcal{G}] - \pi_a) \tag{11.37}$$

which is yet another instance of the delta rule.

The generative model in equations 11.27 and 11.28 is called a mixture of Gaussians model, since it specifies that inputs are drawn from one out of a collection of Gaussians. It finds clusters in the data, since the individual Gaussians are just like individual clusters. Although we have not demonstrated it explicitly, this model tends to generate sparse representations. If the data really are clumped as in a mixture of Gaussians generative model, we can expect that $P[v_a = 1|\mathbf{u};\mathcal{G}] \sim 0$ for most inputs that do not come from cluster a, and $P[v_a = 1|\mathbf{u};\mathcal{G}] \sim 1$ for inputs that do.

A mixture of Gaussians statistical model lies at the heart of the elastic net model that we discuss in chapter 9 as an example of a feature-based competitive model for activity development. The first term in equation 9.19 is exactly the negative average log likelihood of the inputs (from different eyes e and different spatial positions) under a mixture of Gaussians model with equal mixing proportions ($\pi_a = 1/n_v$), where each Gaussian is centered on the position of one cortical points in input space. The second term in the equation expresses the prior expectation that nearby cortical units should map to nearby locations in input space, and plays the key role in ensuring well ordered maps.

11.6 Sparse Coding

The constraint in the mixture of Gaussians architecture is that the final representation of an input $\mathbf{u}$ is a distribution over which *single* unit among the $\mathbf{v}$ should be active. This is much too restrictive – large scale inputs are unlikely to have just a single cause. As an alternative, Olshausen & Field (1996; 1997) suggested a non-linear version of factor analysis for which the distribution of $\mathbf{u}$ given $\mathbf{v}$ is just as before $\mathcal{N}[\mathbf{G}\mathbf{v}, \tau^2\mathbb{I}]$, but the prior distribution over $\mathbf{v}$ is:

$$P[\mathbf{v};\mathcal{G}] \propto \prod_a e^{-f(v_a)} \tag{11.38}$$

where $f(v)$ is a function such as $f(v) = \alpha v$ (an exponential distribution inspired by activity distributions such as that in figure 11.2) or

$$f(v) = \alpha \log(v_0^2 + v^2) \tag{11.39}$$

for some appropriate constants α and v_0. The form in equation 11.39 makes the v_a mutually independent (since the overall probability is a product of the individual probabilities), and gives each a Cauchy distribution. This distribution encourages sparsity, since there is strong pressure for v_a to be small, but only a relatively slowly growing penalty as v_a gets very large. We set $\tau^2 = 1$ for convenience.

If $f(v_a)$ was proportional to v_a^2, $P[\mathbf{v};\mathcal{G}]$ in equation 11.38 would be Gaussian, and the whole system would just perform factor analysis. For arbitrary $f(v_a)$, such as equation 11.39, it is tricky to compute the recognition

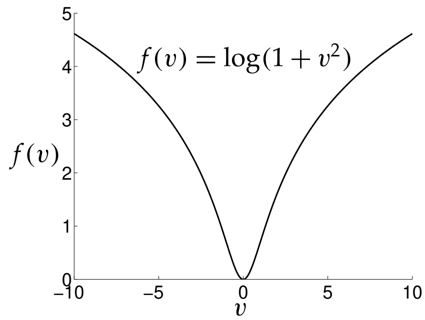

Figure 11.3: Olshausen & Field's (1996) sparseness function $f(v) = \log(1+v^2)$ (*ie* $v_0 = 1$). The sharp dip at $v = 0$ encourages small values for v, and the slow growth means that large values are not penalised too greatly.

distribution $P[\mathbf{v}|\mathbf{u};\mathcal{G}]$ exactly. Olshausen & Field used a deterministic approximation to the recognition distribution:

$$\mathcal{Q}[\mathbf{v};\mathbf{u},\tilde{\mathbf{v}}] = \delta(\mathbf{v} - \tilde{\mathbf{v}}) \tag{11.40}$$

where $\tilde{\mathbf{v}} = \tilde{\mathbf{v}}(\mathbf{u})$ is a free parameter, determined by minimising function $\mathcal{F}[\mathbf{u},\mathcal{Q};\mathcal{G}]$. With this deterministic $\mathcal{Q}[\mathbf{v};\mathbf{u},\tilde{\mathbf{v}}]$ the free energy is just

$$\mathcal{F}\left[\mathbf{u},\tilde{\mathbf{v}};\mathcal{G}\right] \equiv \mathcal{F}[\mathbf{u},\mathcal{Q};\mathcal{G}] = \frac{1}{2}\left|\mathbf{u} - \mathbf{G}\tilde{\mathbf{v}}\right|^2 + \sum_a f(\tilde{v}_a) + \mathbf{K} \tag{11.41}$$

where $\mathbf{K}$ is a constant with respect to $\mathbf{G}$ and $\tilde{\mathbf{v}}$, abusing notation slightly. As in the general scheme, density estimation can be performed by alternating minimisation with respect to $\tilde{\mathbf{v}}$ for a particular input $\mathbf{u}$ and then with respect to $\mathbf{G}$ based on the minimising $\tilde{\mathbf{v}}^*$.

One can seek to find a good value of $\tilde{\mathbf{v}}$ for a given case by using gradient descent for each presented $\mathbf{u}$:

$$\begin{aligned} \tilde{\mathbf{v}} &\rightarrow \tilde{\mathbf{v}} - \epsilon\nabla_{\tilde{\mathbf{v}}}\mathcal{F}[\mathbf{u},\tilde{\mathbf{v}};\mathcal{G}] && (11.42)\\ &= \tilde{\mathbf{v}} + \epsilon\left[\mathbf{G}^T(\mathbf{u} - \mathbf{G}\tilde{\mathbf{v}}) - \mathbf{f}'(\tilde{\mathbf{v}})\right] && (11.43)\\ &= \tilde{\mathbf{v}} + \epsilon\mathbf{G}^T\mathbf{u} - \epsilon\left(\mathbf{G}^T\mathbf{G}\tilde{\mathbf{v}} + \mathbf{f}'(\tilde{\mathbf{v}})\right) && (11.44) \end{aligned}$$

where $[\mathbf{f}'(\tilde{\mathbf{v}})]_a = f'(\tilde{v}_a)$. We assume that this gradient descent occurs quickly compared with the time that a single input case $\mathbf{u}$ is presented. Therefore, we assume that subsequent processing in the network and particularly learning happen on the basis of the equilibrium or stationary values of these equations. For the case of inputs that themselves are changing rapidly, such as visual motion, this may be an unreasonable simplification. Since $\mathcal{F}[\mathbf{u},\tilde{\mathbf{v}};\mathcal{G}]$ is usually not convex in $\tilde{\mathbf{v}}$, there is no guarantee of a unique minimum.

The form of the update in equation 11.43 suggests the network shown in figure 11.4A, in which the activation rule at the input layer requires the

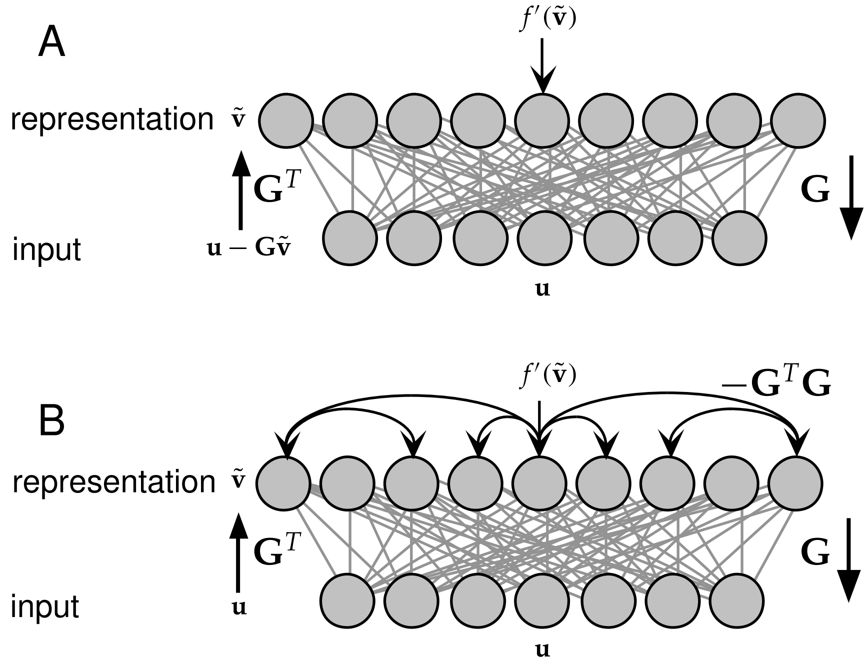

Figure 11.4: Two forms of the sparse coding network. A) This implements equation (11.43) in which the current guess $\tilde{\mathbf{v}}$ is fed back to the inputs through $\mathbf{G}$, and the error $\mathbf{u} - \mathbf{G}\tilde{\mathbf{v}}$ is propagated forward through the transpose of the generative weights $\mathbf{G}^T$ and incorporated with the term from the prior $\mathbf{f}'(\tilde{\mathbf{v}})$. B) This implements equation (11.44), using recurrent weights $-\mathbf{G}^T\mathbf{G}$ in the $\tilde{\mathbf{v}}$ layer to avoid the need to pass $\tilde{\mathbf{v}}$ back to the inputs. Both versions use bottom-up weights which are the transpose of the top-down weights.

calculation of the prediction error $\mathbf{u} - \mathbf{G}\tilde{\mathbf{v}}$ between the actual input and the top-down prediction of the input, and in which this prediction error is propagated bottom-up through weights $\mathbf{G}^T$ which are the transpose of the top-down weights $\mathbf{G}$. The form in equation 11.44 suggests instead sending the input $\mathbf{u}$ directly through the same bottom-up weights, and using symmetric recurrent weights $-\mathbf{G}^T\mathbf{G}$ between the representational units to calculate the term $-\mathbf{G}^T\mathbf{G}\tilde{\mathbf{v}}$. Figure 11.4B shows the resulting network.

Having found the stable value $\tilde{\mathbf{v}}^*$, it is then necessary to alter the weights $\mathcal{G}$ to improve the generative model. Performing gradient descent on $\mathcal{F}[\mathbf{u}, \mathcal{Q}; \mathcal{G}]$, we get

$$\begin{aligned} \mathbf{G}_{ba} &\rightarrow \mathbf{G}_{ba} - \epsilon \nabla_{\mathbf{G}_{ba}} \mathcal{F}[\mathbf{u}, \tilde{\mathbf{v}}^*; \mathbf{G}] && (11.45) \\ &= \mathbf{G}_{ba} + \epsilon [\mathbf{u} - \mathbf{G}\tilde{\mathbf{v}}^*]_b \, \tilde{v}_a^* . && (11.46) \end{aligned}$$

where ϵ is the learning rate. This learning rule is again just the delta rule, where adaptation is driven by the reconstruction error $\mathbf{u} - \mathbf{G}\tilde{\mathbf{v}}^*$.

Note that the prior over v_a, $e^{-f(v_a)}$ only directly plays a part in determining the activities $\tilde{\mathbf{v}}^*$, not in setting the weights. Various extra constraints are also required. For instance, low activities $\mathbf{v}$ could be simply arranged by having large weights $\mathbf{G}$. This is prevented by normalising the weights. Also, an equalisation constraint is employed to ensure that all units are roughly equally active.

When this activation and learning rule are applied to inputs based on (prewhitened) intensities of small square patches of monochrome photographs of natural scenes, it develops selectivities for the v_a that quite closely resemble those of cortical simple cells, with a population of oriented line detectors (similar to Gabor functions) at various scales. Figure 11.5A shows the weights $\mathbf{G}$ that develop in this case, one picture for each representational unit. Note that these weights are actually the *projective* rather than the receptive fields of the units, since they govern the distribution of $\mathbf{u}$ given $\mathbf{v}$. There is no simple form of receptive field for the units, since $\tilde{\mathbf{v}}$ is not determined by a feedforward equation such as $\tilde{\mathbf{v}} = \mathbf{W}\mathbf{u}$ but rather by minimising the expression in equation 11.44. One can *map* a receptive field by presenting different inputs and recording the response, but its precise form depends on the inputs used to do the mapping. Figure 11.5B shows an example of this – the receptive field predicted from the gratings shows ringing since the frequency response has a sharp cut-off owing to the sparsification.

This sparse-coding network demonstrates several important points. First, it shows the way that priors over the causes $\mathbf{v}$ play a major role in specifying what sort of causes are extracted. Here, the priors encourage the extraction of sparse and independent causes. By contrast, since the input patterns used to develop the projective fields in figure 11.5A were prewhitened, PCA would have failed to report anything interesting at all. The principal components even of the unwhitened data do not show the localised Gabor-like structures of this figure, but rather are blurred and coarse. Second, the sparse-coding network shows the interaction between the two phases of the algorithm we have discussed for performing ML density estimation – first finding the appropriate representation $\tilde{\mathbf{v}}^*$ and then using it to specify learning. Third, it uses the same very simple approximation for the recognition distribution that we used above in the analysis of PCA. Fourth, although under the prior, the activities of $\mathbf{v}$ are mutually independent, in fact they form something more akin to a distributed population code for the spatial structure of the input, and are likely to have significant interdependencies. We study these interdependencies below in the context of more systematic forms of multi-resolution analysis models. To put this the other way around, population codes, whose activities are rarely independent, are neurally ubiquitous, so it is unfortunate for there to be pressure from the prior that acts *against* creating them. Finally, the fact that the causes extracted are like little Gabor patches should be worrying, since it is at the very least unnatural to conceive of images as only having such low level causes, rather, for instance, than causes couched in terms of the objects contained in the images. From the perspective of good representations, causes that are more like objects and less like Gabor patches would be more useful. It is somewhat more

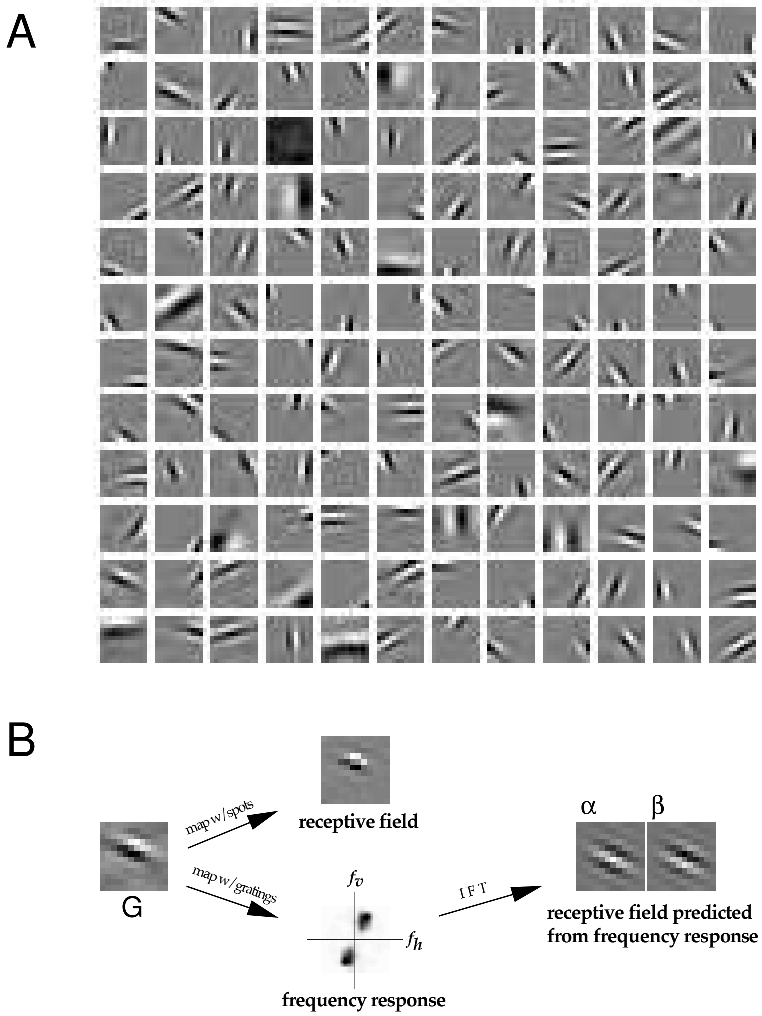

Figure 11.5: Projective and receptive fields for the sparse coding network. A) Projective fields $\mathbf{G}_{ab}$ (a is an index over representational units $\mathbf{v}$; b is an index over input units $\mathbf{u}$) of 144 units for a case in which $\mathbf{u}$ is a 12×12 pixel grid. The weights are plotted in grey-scale (grey is 0) in the same shape as the input grid. These projective fields form a Gabor-like representation for images, covering a variety of spatial scales and orientations. B) The relationship between projective and receptive fields. The picture on the left shows a projective field of one of the units. The middle column shows its responses mapped either as activity in response to dot inputs (top) or its frequency response to gratings. Performing an inverse Fourier transform (IFT) of the frequency response, assuming zero phase for all the components leads to the spatial receptive field shown in α (β shows the Hilbert transform of α, which can be used to show the effects of changing the phase). Note that the proportion of theq units that favor higher spatial frequencies is rather greater than for simple cells in V1. From Olshausen & Field (1997).

reasonable to conceive of causes in a hierarchical manner, using causes at each level to account for correlations amongst the causes at lower levels. However, such hierarchical models are still very rudimentary.

11.7 Independent Components Analysis

The overall probability distribution for the sparse-coding network has

$$P[\mathbf{u},\mathbf{v};\mathcal{G}] \propto \left[\prod_a e^{-f_a(v_a)}\right] e^{-|\mathbf{u}-\mathbf{G}\mathbf{v}|^2/2\tau^2} \tag{11.47}$$

Consider a special case of this for which $\mathbf{G}$ is invertible, with $\mathbf{W}=\mathbf{G}^{-1}$ (this requires $\mathbf{u}$ and $\mathbf{v}$ to have the same dimensionality), $\tau^2=0$, and the v_a are independent, and all have different prior distributions $P_a[v_a;\mathcal{G}] = p_a(v_a) = e^{-f_a(v_a)}$. Then,

$$P[\mathbf{u},\mathbf{v};\mathcal{G}] = \left[\prod_a e^{-f_a(v_a)}\right] \delta(\mathbf{u}-\mathbf{G}\mathbf{v}) \tag{11.48}$$

Since for scalars, $\int_u \delta(u-gv)p(v)dv = \frac{1}{g}p\left(\frac{u}{g}\right)$, the likelihood of an example can be written as:

$$\begin{aligned}\log P[\mathbf{u};\mathcal{G}] &= \log \int_{\mathbf{v}} \left[\prod_a p_a(v_a)\right] \delta(\mathbf{u}-\mathbf{G}\mathbf{v})d\mathbf{v} && (11.49)\\ &= \log \frac{1}{|\mathbf{G}|}\prod_a p_a\left([\mathbf{W}\mathbf{u}]_a\right) && (11.50)\\ &= \log|\mathbf{W}| + \sum_a \log p_a\left([\mathbf{W}\mathbf{u}]_a\right) && (11.51)\end{aligned}$$

where $|\mathbf{W}| = 1/|\mathbf{G}|$ is the determinant of $\mathbf{W}$.

Considering this in terms of $\mathbf{W}$, direct gradient ascent on this log likelihood can be performed using:

$$\begin{aligned}\mathbf{W}_{ab} &\rightarrow \mathbf{W}_{ab} + \epsilon\frac{\partial}{\partial \mathbf{W}_{ab}} \log P[\mathbf{u};\mathbf{W}^{-1}] && (11.52)\\ &= \mathbf{W}_{ab} + \epsilon\left(\left[\mathbf{W}^T\right]^{-1}_{ab} + z_a u_b\right) && (11.53)\end{aligned}$$

where $z_a \equiv \frac{\partial}{\partial v_a}\log p_a(v_a)$, since $\partial/\partial\mathbf{W}_{ab}\left(\log|\mathbf{W}|\right) = \left[\mathbf{W}^T\right]^{-1}_{ab}$. Consider the special case in which $p_a(v_a) \propto 1/\cosh(v_a)$, then $z_a = -\tanh(v_a)$, and so

$$\mathbf{W}_{ab} \rightarrow \mathbf{W}_{ab} + \epsilon\left(\left[\mathbf{W}^T\right]^{-1}_{ab} - \tanh(v_a)u_b\right) \tag{11.54}$$

is the sum of an anti-weight decay term that forces $\mathbf{W}$ to be non-singular and an anti-Hebbian rule (the weight decreases if u_b and v_a are jointly active). Since $\mathbf{W}^T\mathbf{W}$ is a positive definite matrix, the weight change in equation 11.54 can be multiplied by $\mathbf{W}^T\mathbf{W}$ without affecting the fixed points. This leaves

$$\mathbf{W}_{ab} \rightarrow \mathbf{W}_{ab} + \epsilon \left(\mathbf{W}_{ab} - \tanh(v_a) \left[\mathbf{v}^T \mathbf{W} \right]_b \right) \tag{11.55}$$

as the net weight change. This avoids the need for inverting $\mathbf{W}$, although at the expense of requiring the calculation of $\mathbf{v}^T\mathbf{W}$.

The resulting algorithm is called an independent components algorithm. It is a non-linear generalisation of PCA that places its stress on making the output components mutually independent, and thereby extracting independent causes (*ie* components) from the input. This architecture makes a strong underlying linearity assumption that $\mathbf{u} = \mathbf{G}\mathbf{v}$. There are also some other technical conditions that must be satisfied for it to work, specially that the distributions $p_a(v_a)$ be non-Gaussian and also faithfully reflect the actual distribution over the causes.

The algorithm was originally suggested by Bell & Sejnowski (1995) from a different perspective. They started from Linkser's idea of maximising the mutual information between $\mathbf{u}$ and $\mathbf{v}$, only in a non-linear deterministic network where $v_a(\mathbf{u}) = \sigma\left([\mathbf{W}\mathbf{u}]_a\right)$ rather than the linear deterministic network we considered in chapter 8 that led to PCA. Maximising the mutual information in this context requires maximising the unconditional entropy $\mathcal{H}[\mathbf{v}]$. For any invertible mapping $\mathbf{v} = \mathbf{f}(\mathbf{u})$, the probability density of $\mathbf{v}$ is given by $p_{\mathbf{V}}(\mathbf{v}) = \frac{p_{\mathbf{U}}(\mathbf{f}^{-1}(\mathbf{v}))}{|\mathbf{J}(\mathbf{f}^{-1}(\mathbf{v}))|}$, where $p_{\mathbf{U}}(\mathbf{u})$ is the probability density of $\mathbf{u}$, and $\mathbf{J}(\mathbf{u})$ is the Jacobian of the mapping $\mathbf{J}(\mathbf{u})_{ab} = \partial v_a(\mathbf{u})/\partial u_b$. In this case, writing $v'(u) = d\sigma(u)/du$ we have $|\mathbf{J}(\mathbf{u})| = |\mathbf{W}| \prod_{a=1}^{n_v} |v'_a(\mathbf{W}\mathbf{u})|$. The main term in the entropy that changes with $\mathbf{W}$ is

$$\langle \log |\mathbf{J}(\mathbf{u})| \rangle_{P_I[\mathbf{u}]} = \log |\mathbf{W}| + \sum_a \log |v'_a(\mathbf{W}\mathbf{u})| \tag{11.56}$$

So, if $v'_a(u) = p_a(u)$, then equations 11.56 and 11.51 are identical. Note that the two procedures of *minimising* the summed bit entropies of the output subject to a non-triviality constraint (which is discussed in chapter 4) and *maximising* the mutual information between $\mathbf{u}$ and $\mathbf{v}$ by maximising the joint entropy of $\mathbf{v}$ have the same effect here of finding factorial representations. However, not all methods for finding such representations can be viewed in terms of density estimation for a causal model.

Figure 11.6 shows an example due to Bell & Sejnowski (1996) of independent components analysis applied to sounds. In this case, the input was created by tapping a tooth whilst adjusting the shape of the mouth, and it was sampled from a microphone at 8kHz (figure 11.6A;B;C). A close variant of the independent component analysis method described was used on $10,000$ samples of length 100 (1/80 seconds) chosen at random times from the overall input, leading to 100 independent components.

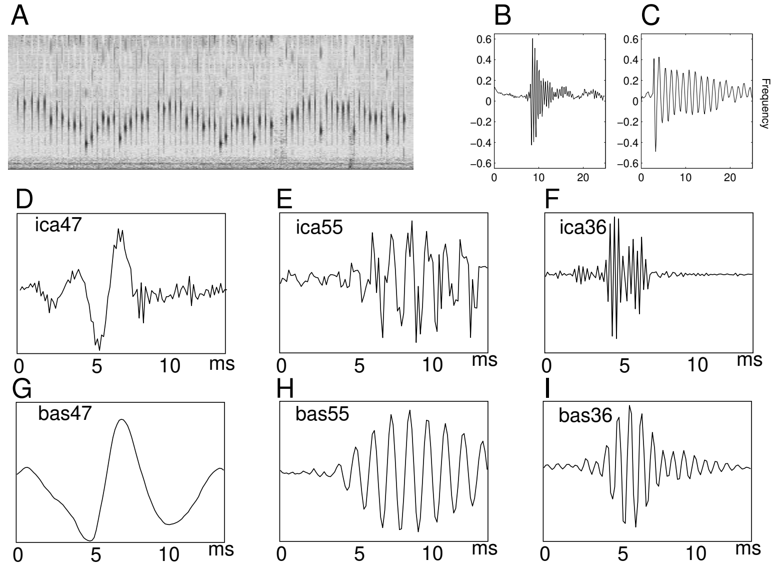

Figure 11.6: Independent components of tooth-tapping sounds. A) Spectrogram of a segment of a tune by Beethoven played by tapping a tooth. B;C) waveforms of taps for high and low frequency notes respectively. The mouth acts as a damped resonant cavity. D;E;F) Three independent components calculated on the basis of 100 samples (1/80 seconds) generated at random time points from the data – the graphs show the receptive fields $\mathbf{W}_{i*}$ for three output units. D is reported as being sensitive to the sound of the air-conditioner; E;F extract tooth taps of two different frequencies. G;H;I) are the associated rows of $\mathbf{G} = \mathbf{W}^{-1}$, showing what activity in the input is caused by activity in D;E;F. Adapted from Bell & Sejnowski (1996).

Figure 11.6D;E;F show receptive fields ($\mathbf{W}$) of three of the components – the last two in particular extract particular frequencies in the input. Figure 11.6G;H;I show the projective fields ($\mathbf{G}$) of these units showing what the consequence in the input is of activating these causes. Note that the projective fields are much smoother than the receptive fields – generation can be seen as more cooperative than recognition, since the task for recognition is dividing responsibility for the input between the various possible causes.

Independence is an important characteristic of causes. However, the assumption of linear mixing is too strong for natural inputs, as is the forced equality of dimension between input and output. The requirement that the $p_a(v)$ be the distribution functions of the original sources is also questionable. However, these priors may themselves be adaptive, an interpretation we see later in the slightly different context of the Helmholtz machine. Also, independent component analysis has been shown to work in a number of cases in which this is clearly false.

Bell & Sejnowski (1997) used similar input data to those used by Olshausen & Field (1996, 1997), and the prior $p_a(v) \propto 1/\cosh(v)$, and showed that

ICA extracted somewhat similar Gabor-like receptive fields as those of the units shown in figure 11.5A. Remember that the actual weights shown in figure 11.5A are actually projective rather than receptive fields. One way to interpret this result is to recall the analysis in chapter 4 about the (second order) decorrelation and whitening of visual inputs. Finding a whitening filter conceptually involves two stages: projecting the input onto the eigenvectors (*ie* the principal components) of covariance matrix of the inputs, and then dividing each output by the square root of the associated eigenvalue. If the inputs had a Gaussian distribution, then this leaves the outputs mutually independent. Furthermore, any further rotation of the outputs (by an orthonormal matrix) also leaves them mutually independent. If the inputs do not have a Gaussian distribution, then further conditions can be applied to favor one particular rotation over others (Li & Atick, 1994). One obvious condition is to make the outputs decorrelated to higher than second order – Bell & Sejnowski's (1997) result for visual input suggests that doing this leads to a Gabor-like basis. We see below that it is possible to go one stage further and remove some of the remaining dependence even after the rotation to this basis.

11.8 Multi-Resolution and Wavelet Models

The weights learned by the sparse coding network shown in figure 11.5 and similarly learnt by ICA applied to the same input data make it perform something akin to a multi-resolution decomposition of its input. In this, each point in the visual world is represented by the activity of a whole population of units which are tuned to widely varying spatial frequencies and orientations centered at particular input locations, being localised in both space and spatial frequency. Systematic versions of such representations, in forms such as wavelets, pyramid codes and quadrature mirror filters, have become extremely important for storing, communicating and manipulating images (and also sounds) on computers, and there is an extensive body of theory about their workings and their representational and coding qualities. Evidence (for instance from studies into adaptation to particular spatial frequencies) shows that such differentiation into separate frequency bands at different spatial locations has significant psychophysical consequences.

Figure 11.7 shows the workings of a fairly stripped-down multi-resolution analysis of a photograph (itself reproduced at the lower left of figure 11.7A). Vertical and horizontal filters (shown in real and Fourier space in figure 11.7B) are applied across the whole image at successively higher spatial scales, leading to the complete decomposition shown in the figure (plus an extra highpass image that is not shown). The tiny image at the top left is the final lowpass image; the remaining images in the 'triangle' show the vertical (upper) and horizontal (lower) structure. Altogether, they can be used to reconstruct the whole photograph almost perfectly.

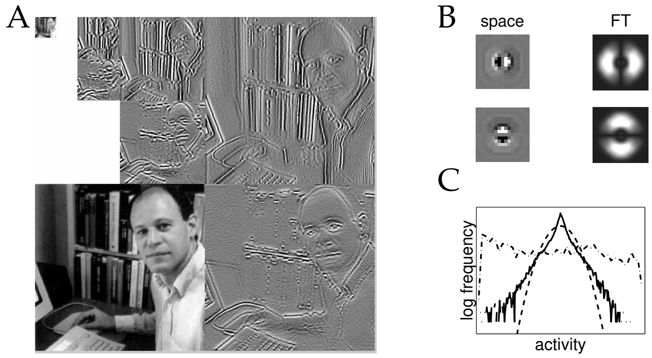

Figure 11.7: Multi-resolution image decomposition. A) A greyscale image (lower left) is decomposed using a pair of vertical (upper in each set) and horizontal (lower in each set) filters into a lowpass image (top left) and three successively higher spatial frequency oriented bandpass images (plus a further highpass image which is not shown). B) Real and Fourier (FT) space versions of the basic vertical and horizontal filters used at multiple positions and spatial scales in (A). C) Log frequency distribution of the activity across one filter sub-band (solid line) compared with that of a Gaussian with the same mean and variance (dashed line) and that of the original image (dot-dashed) line. The pixel values of the image were linearly rescaled to fit into the range. From Simoncelli & Freeman, 1995; Karasaridis & Simoncelli, 1996; 1997.

Figure 11.8 shows the same decomposition as in figure 11.7A, except that all the images are given at the same size to emphasise the separation of different spatial frequencies in the different images. Figure 11.8A is the original image; figure 11.8E is the lowpass image; figure 11.8B;C;D are the vertically oriented bandpass images; and figure 11.8F;G;H are the horizontally oriented bandpass images.

There is a large range of possible multi-resolution decompositions, with a mix of different computational and representational qualities. We will just discuss some basic properties as they relate to representational issues. Almost all the decompositions can be expressed as (complicated) linear transformations $\mathbf{v} = \mathbf{H}\mathbf{u}$ where the rows of $\mathbf{H}$ contain the filters, such as those shown in figure 11.7B. The filters can often be thought of as having similar underlying forms, but being cast at different spatial scales and centered at different locations in input space.

If $\mathbf{H}$ is an orthonormal matrix, satisfying $\mathbf{H}\mathbf{H}^T = \mathbf{I}$, then we can recover the original input by the simple operation $\mathbf{u} = \mathbf{H}^T\mathbf{v}$. Further, if the inputs are decorrelated, with covariance matrix $\langle \mathbf{u}\mathbf{u} \rangle = \mathbf{I}$, then the covariance matrix of the outputs will also be $\mathbf{I}$. In chapter 4, we saw that in Gaussian cases, the orthonormality of $\mathbf{H}$ guarantees that $\mathbf{v}$ can retain optimal information theoretic properties possessed by decorrelated $\mathbf{u}$, such as maximising the information about a signal in the face of noise.

Multi-resolution decompositions throw into sharp focus many of the issues discussed in this chapter. First, consider sparseness. The solid line

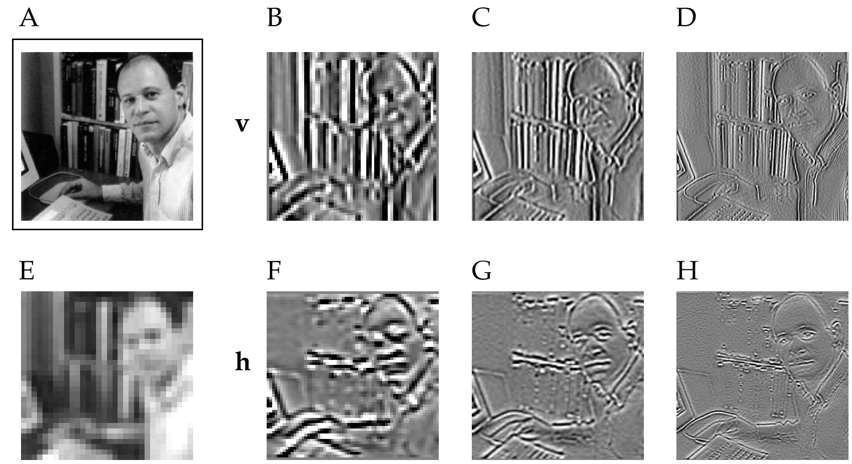

Figure 11.8: Multi-resolution image decomposition. The same decomposition as in figure 11.7 except that all images are shown as being the same size. A) original image; (B;C;D) successively higher spatial frequency vertically oriented bandpass images; E) low pass image; (F;G;H) successively higher spatial frequency horizontally oriented bandpass images.

in figure 11.7C shows the empirical distribution of the activity across all the finest scale, vertically tuned units (the block at the top right of figure 11.7A). Compared with a Gaussian with the same variance (dashed line), the distribution is sparse in the sense that it has higher kurtosis, and is more likely to be either near 0 or have a large magnitude. Over a wide range, its distribution is quite similar to a double exponential, an unrectified version of the spike count distribution of an IT cell shown in figure 11.2. Note that calculating this distribution across the activity of units in different spatial positions for a single image includes an implicit assumption that the statistics of the visual world are translation invariant.

Second, consider multi-resolution analysis as providing a probabilistic model for images. Since optimising the weights of the sparse-coding network led to just such a decomposition, we may expect systematic multi-resolution decompositions to be good image models. The natural way of testing this is to compare the likelihoods afforded by different (but comparably complex) models to collections of images. Another, more indirect, way is to use the probabilistic models to *code* images as well as possible, and evaluate the mdoels according to the the brevity of the codes. A theorem of Shannon guarantees that these ways of comparing models will be the same. Multi-resolution methods (of various sorts) are quite standard for coding images partly because they offer amongst the best, computationally tractable, probabilistic models.

One hint as to the power of multi-resolution decompositions comes from figure 11.7C in the face of a (quite poor) probabilistic model that treats all the units in the decomposition as being independent. Up to the usual additive constants that have to do with the precision with which the activities are coded, the contribution to the cost from a single unit is the entropy of the probability distribution of its activity. The distribution over the activities of the input pixels (whose values have been linearly rescaled to fit into the plot) is shown by the dot-dashed line. This is close to being uniform, which is the maximum entropy distribution for a variable with a fixed range. The sparse distributions of the activities of the units in the multi-resolution decomposition have significantly lower entropy, and therefore can make for cheaper codes and thus models with higher likelihoods.

The perceptual consequences of making the code for an image lossy, by quantising the components of the decomposition representing the highest spatial frequencies very coarsely, are usually rather small. This is an example of how trying to build probabilistic generative models of the input is needlessly tough. Is is likely to be impossible to use abstract principles that are independent of details about the use to which representations are to be turned or the particular domains to decide which information in the input can safely be discarded.

Figures 11.9A;B show one departure from the independence of the units in multi-resolution decompositions that is straightforward to model. Both figures show conditional distributions calculated across a decomposition of horizontally tuned units at the finest scale (child units, C) given the activity of horizontally-tuned units at the next coarser scale (parent units, P). These activities are from a slightly different sort of wavelet decomposition from that in figure 11.7A, involving quadrature mirror filters. Although the *mean* of the distributions $\mathcal{E}[C|P]$ is roughly 0, there is a residual correlation between the magnitude of $|P|$ and the magnitude of $|C|$ (*eg* Shapiro, 1993). Put another way, structure in the image is coordinated across different spatial scales so it is unusual to find high values of $|P|$ at a coarse scale, without some activity at finer scales, albeit activity whose sign may be difficult to predict from . Following Simoncelli (1997), figure 11.9B shows this more directly by showing the conditional distributions of $\log|C|$ given $\log|P|$. For small values of $\log|P|$, perhaps below a noise cut-off, the distribution of $\log|C|$ is flat; for larger values of $\log|P|$, the growth in the variance of C is clear.

It is straightforward to build a better probabilistic model of images by learning the dependence shown in figure 11.9A;B. In general, similar statistical dependencies exist between different orientations and across more widely separated spatial scales. These can also be taken into account. In terms of the sparse network or ICA, this amounts to more sophisticated models than equations 11.38 or 11.48 in which the activities of the units $\mathbf{v}$ are not independent but covary in particular ways. Hierarchical probabilistic models such as the Helmholtz machine, which is discussed below, also attempt to model such covariation.

An alternative to attempting to model this statistical independence is attempting to remove it. Simoncelli & Schwartz (1999) suggest using a form of divisive normalisation in which each unit is part of a pool of units, each

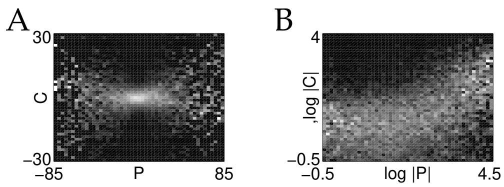

Figure 11.9: A) Gray-scale coded conditional distribution of the response of a child unit (C) at the finest scale of a three-scale quadrature mirror filter decomposition of the same picture as in figure 11.7A as a function of the magnitude of its parent (P) at the same position and orientation, but coarser spatial scale. Each column is separately normalised. The plot has a characteristic bow-tie shape. B) Re-representation of the same data, as the conditional distribution of $\log|C|$ as a function of $\log|P|$. After Simoncelli & Adelson, 1990; Simoncelli & Schwartz, 1999.

of which helps normalise the others. Divisive normalisation has been suggested by Heeger (1992) to account for empirical results on contrast normalisation. Such normalisation is something of a departure from the general scheme in this chapter of finding representations by building probabilistic models of the input, since, if successful, it produces somewhat factorial representations of the input which cannot be expected to be a good basis for reconstructing the input, since information on the absolute rather than the relative magnitudes of the components has been normalised away and therefore lost.

A further aspect brought home by multi-resolution representations is completeness. The discussion so far has not treated the issue of whether there might be more output units than input units – such representations are called *over-complete.* This is important as far as the cortex is concerned since, for instance, there are around 40 times as many cells in V1 as inputs from the LGN, suggesting that there is a large representational expansion. It also suggests that units in V1 might well *not* be independent. Although over-complete representations require more output units, properties of easy invertibility and appropriate coding can often be preserved by making the decomposition matrix $\mathbf{H}$ form a *tight frame, ie* to have $\mathbf{H}^T\mathbf{H} = \mathbf{I}$. This implies, for instance, that $\mathbf{u} = \mathbf{H}^T\mathbf{v}$, just as if $\mathbf{H}$ were orthonormal.

Many possible reasons have been suggested for over-completeness – although none obviously emerges from the requirement of fitting good probabilistic models to the input. One important idea comes from the notion that the task of manipulating representations should be invariant to groups of transformations of the input – for images, at least rotation, translation and scaling. For instance, although the image decomposition in figure 11.7A is designed to be *steerable,* in the sense that one can calculate the responses of filters at orientations other than horizontal and vertical within a scale by simple linear combinations of the responses of the horizontally and vertically tuned units within the same scale, the representation of other orientations is *implicit* rather than *explicit.* This means, for instance, that the operations required to manipulate contours would be different if the contours are presented at different orientations (at the very least requiring a steering operation). Since orientations are explicitly represented roughly equally in V1 (bar a slight bias for horizontal and vertical), the operations required to manipulate contours are more nearly the same across different orientations. The over-completeness of the cortical representation is a necessary consequence. In more realistic models that include the possibility of noise in the activities of the units, other rationales for over-completeness come from considering representations in terms of population codes, in which many units redundantly report information about closely related quantities so that uncertainty can be reduced.

Similarly, the computational task of recognising objects in the face of translation and scaling, two operations that happen naturally to visual inputs as we manoeuvre around the objects that create them, would be significantly eased given a representation of the visual input that has a measure of invariance to these manipulations. In cases for which the weights for each output *within* a particular scale are translations of each other and *between* scales are self-similar, the effect of a pure spatial translation of the input by an amount equal to the granularity of sampling at that scale is to shift a single pattern of activity across the units and the effect of a pure scaling of the input by an amount equal to the scale factor between levels of the decomposition is to shift the same pattern of activity onto a different of units. Such a set of filters is possible with an orthonormal $\mathbf{H}$, although doing so in the case that the scale factor between spatial scales is a cortically reasonable factor of 3 requires there to be at least two different classes of cells at each scale, for instance ones tuned to different orientations (Li & Atick, 1994). However, imposing additional requirements on the stability of the representation as the input translates and scales continuously by less than the granularity at any level imposes stringent conditions on the decomposition that can only be (even approximately) satisfied in the face of significant over-completeness in the representation.

Multi-resolution decompositions therefore illuminate many of the representational issues that are central in this chapter, such as sparseness and independence, whilst themselves being the natural outcome of some prominent learning models. The most important lacuna they reveal is the inadequacy of judging the quality of models merely by the likelihood, since representations are formed in the service of computations rather than mere coding. For instance, over-completeness in orientation space is usually irrelevant from the perspective of the likelihood, but is highly relevant given a particular scheme by which computations may be orientation invariant. Understanding how to incorporate such constraints into the design of likelihood models is key.

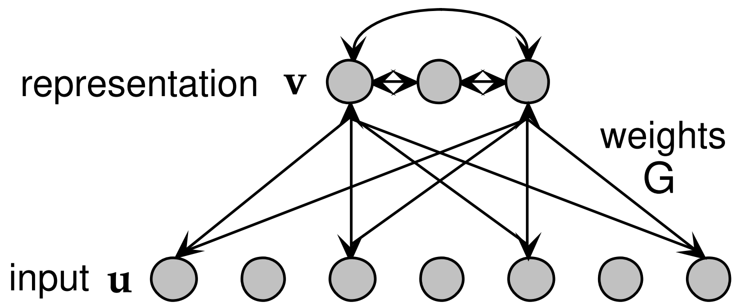

Figure 11.10: The Boltzmann machine. The Boltzmann machine can have symmetric bidirectional weights **G** between all units, although it need not be fully connected. **u** and **v** are both binary, and the representation **v** for a given **u** is determined by Gibbs sampling.

11.9 The Boltzmann Machine

The Boltzmann machine was the first network that was explicitly designed to perform density estimation. It consists of binary stochastic units (u_a, $v_{a'} \in \{0, 1\}$) connected by bidirectional weights $\mathbf{G}=\mathbf{G}^T$. In general the units may also have biases, which, as usual, we treat as weights from a unit whose value is always 1. It is important that there are no self-weights, *ie* $\mathbf{G}_{aa}=0$ for all a. All the weights actually form part of the generative model (so in this case, $\mathcal{G} = \mathbf{G}$). Figure 11.10 shows the architecture of a version of the Boltzmann machine that has been designed to be like the generic ML density estimation architecture shown in figure 11.1. Arbitrary (symmetric) connection schemes are also possible, including ones with multiple layers in the hierarchy.

Rather than decomposing $P[\mathbf{u}, \mathbf{v}; \mathcal{G}]$ as in expressions 11.1 and 11.2, the Boltzmann machine specifies the generative probabilities through an energy function. If **u** are the (binary) states of the input or visible units, **v** are the (binary) states of the hidden or representation units, and, for convenience, $\mathbf{z} \equiv \{\mathbf{u}, \mathbf{v}\}$ are the states of all the units (we write z_k for a generic element of **u** or **v**), the energy function of the Boltzmann machine is defined as:

$$E(\mathbf{z}; \mathbf{G}) = -\frac{1}{2}\mathbf{z}^T\mathbf{G}\mathbf{z}. \tag{11.57}$$

Inspired by a physical model in which units are like particles, this energy function is used to define a probability distribution over **z**

$$P[\mathbf{z}; \mathbf{G}] = \frac{1}{\mathcal{Z}(\mathbf{G})} e^{-\beta E(\mathbf{z};\mathbf{G})} \tag{11.58}$$

where

$$\mathcal{Z}(\mathbf{G}) = \mathcal{Z} = \sum_{\mathbf{z}} e^{-\beta E(\mathbf{z};\mathbf{G})} \tag{11.59}$$

is a normalizing constant (called the *partition function* in physics), and β is the analogue of an inverse temperature. This distribution does not have any *intrinsic* meaning – there is no *a priori* relationship between the activities of the states and $P[\mathbf{z};\mathbf{G}]$. Rather this is something that is arranged by the dynamical behavior of the network. Note, as in equation 11.3,

$$P[\mathbf{u};\mathbf{G}] = \sum_{\mathbf{v}} P[\mathbf{u},\mathbf{v};\mathbf{G}] = \sum_{\mathbf{v}} P[\mathbf{z};\mathbf{G}] \tag{11.60}$$

and, once again, we interpret $\mathbf{v}$ as representing $\mathbf{u}$ in terms of its possible causes.

Before describing the dynamics, we derive the gradient learning rule for setting the weights of the Boltzmann machine. Although we could use the general analysis for hidden cause models, it is straightforward to find the gradient of the log likelihood directly. Without loss of generality for learning (though not necessarily for the dynamical behavior of the network), we can set $\beta = 1$, since β only determines the scale of the connections $\mathbf{G}_{ij}$. Note first that:

$$\nabla_{\mathbf{G}_{ij}} P[\mathbf{z};\mathbf{G}] \propto P[\mathbf{z};\mathbf{G}] \left(z_i z_j - \sum_{\mathbf{z}'} P[\mathbf{z}';\mathbf{G}] z'_i z'_j \right) \tag{11.61}$$

Then, since $\log P[\mathbf{u};\mathbf{G}] = \log \sum_{\mathbf{v}} P[\mathbf{u},\mathbf{v};\mathbf{G}]$, we have that:

$$\nabla_{\mathbf{G}_{ij}} \log P[\mathbf{u};\mathbf{G}] = \nabla_{\mathbf{G}_{ij}} \log \sum_{\mathbf{v}} P[\mathbf{u},\mathbf{v};\mathbf{G}] \tag{11.62}$$

$$= \frac{\sum_{\mathbf{v}} \nabla_{\mathbf{G}_{ij}} P[\mathbf{z};\mathbf{G}]}{P[\mathbf{u};\mathbf{G}]} \tag{11.63}$$

$$\propto \sum_{\mathbf{v}} P[\mathbf{v}|\mathbf{u};\mathbf{G}] z_i z_j - \sum_{\mathbf{z}'} P[\mathbf{z}';\mathbf{G}] z'_i z'_j \tag{11.64}$$

The first term is an average over the recognition distribution:

$$P[\mathbf{v}|\mathbf{u};\mathbf{G}] = \frac{P[\mathbf{z};\mathbf{G}]}{P[\mathbf{u};\mathbf{G}]} = \frac{e^{-E(\mathbf{v},\mathbf{u};\mathbf{G})}}{\sum_{\mathbf{v}'} e^{-E(\mathbf{v}',\mathbf{u};\mathbf{G})}} \tag{11.65}$$

and the second is an average over the total distribution of all the units in the network. Unfortunately, neither of these distributions is tractable to compute, since if there are n_u input units $\mathbf{u}$ and n_v representational units $\mathbf{v}$, the sum in the conditional distribution is over 2^{n_v} terms, and the sum in the unconditional distribution is over $2^{n_u+n_v}$ terms.

One strategy for learning is to create dynamics for the network that generate *samples* from these distributions. For the first term, given samples $\mathbf{v}^\nu$ for $\nu = 1 \ldots K$ from the distribution $P[\mathbf{v}|\mathbf{u};\mathbf{G}]$ (and writing $\mathbf{z}^\nu = \{\mathbf{u},\mathbf{v}^\nu\}$), we can approximate

$$\sum_{\mathbf{v}} P[\mathbf{v}|\mathbf{u};\mathbf{G}] z_i z_j \sim \frac{1}{K} \sum_{\nu=1}^{K} z_i^\nu z_j^\nu \equiv \langle z_i z_j \rangle_+ . \tag{11.66}$$

For the second term, given samples $\mathbf{z}^{'\nu}$ for $\nu = 1 \ldots K$ from the unconditional distribution $P[\mathbf{z}'; \mathbf{G}]$, we can approximate

$$\sum_{\mathbf{z}'} P[\mathbf{z}'; \mathbf{G}] z'_i z'_j \sim \frac{1}{K} \sum_{\nu=1}^{K} z_i^{'\nu} z_j^{'\nu} \equiv \langle z'_i z'_j \rangle_- . \tag{11.67}$$

The method by which the Boltzmann machine produces samples of either distribution is called *Gibbs* sampling, and is described in appendix B. Gibbs sampling involves a Markov chain whose stationary distribution is set to be the distribution from which samples need to be taken. In this case, sampling involves nothing more difficult than a stochastic form of the asynchronous update of a Hopfield net:

$$p[z_k = 1 | \mathbf{z}_{\bar{k}}; \mathbf{G}] = \sigma \left(\sum_{k' \neq k} \mathbf{G}_{kk'} z'_k \right) \tag{11.68}$$

where

$$\sigma(c) = \frac{1}{1 + e^{-c}} \tag{11.69}$$

is the standard sigmoid function, and $\mathbf{z}_{\bar{k}}$ consists of the binary states of all units other than unit k. States k are updated according to equation 11.68 one at a time, chosen either systematically or at random.

Another strategy for approximating these expectations is to use mean field approximations to the relevant distributions, as described in appendix C. Mean field methods choose a simple parameterised approximation to the distribution, and optimise the parameters to make the approximation as tight as possible. If the approximation has an appropriate form, then the exact expectations for the approximate distributions are easy to calculate. In this case, the appropriate simplification is to model all the units in the network as being independent, and one can then use a continuous form of the dynamics of a Hopfield network (see chapter 8) to optimise the parameters. The net result is equivalent approximations for $\langle z_i z_j \rangle_+$ and $\langle z'_i z'_j \rangle_-$.

In either case, gradient ascent in the log likelihood is performed by changing $\mathbf{G}_{ij}$ according to

$$\mathbf{G}_{ij} \to \mathbf{G}_{ij} + \epsilon \left(\langle z_i z_j \rangle_+ - \langle z'_i z'_j \rangle_- \right) . \tag{11.70}$$

In the case of Gibbs sampling, this is stochastic gradient ascent in two senses. First, the input patterns $\mathbf{u}$ are generated stochastically from $P_I[\mathbf{u}]$. Second, the representations $\mathbf{v}^\nu$ of the input $\mathbf{u}$ are themselves stochastic samples from the underlying distribution $P[\mathbf{v}|\mathbf{u}; \mathcal{G}]$.

The production of $\langle z_i z_j \rangle_+$ is (rather metaphorically) called the *wake* phase, since it is based on the observed pattern $\mathbf{u}$. It involves Hebbian learning of the connections. The production of $\langle z'_i z'_j \rangle_-$ is called (equally metaphorically) the *sleep* phase, since it happens in the absence of any particular input pattern. Sleep phase learning involves an anti-Hebbian rule and has also been characterised as *unlearning.* Learning stops when the correlations during the wake phase equal the correlations during the sleep phase, at which point the network will be at a local maximum of the likelihood. There is no need to normalise or limit the weights, since the Hebbian and anti-Hebbian learning balance each other in an exactly appropriate manner.

The Boltzmann machine is theoretically highly attractive. Given enough representational units $\mathbf{v}$, it can represent any probability distribution $P_I[\mathbf{u}]$ with arbitrary accuracy. Further, the learning rule only requires the extremely simple and local Hebbian and anti-Hebbian learning rules without even any synaptic normalisation. Even the requirement for bidirectional and symmetrical weights is not necessarily a large concern – for instance, top-down weights invariably accompany bottom-up weights in the canonical cortical macrocircuit (see chapter 7), and the learning rule can be slightly modified to encourage the Boltzmann machine to produce symmetric weights.

Unfortunately, the Boltzmann machine does not seem to work well in practice. Gibbs sampling is notoriously inefficient as a way of expressing probability distributions, particularly during the sleep phase in which the network is completely unclamped. Furthermore, learning is based on the difference between two noisy signals – and so the signal to noise ratio in the direction of weight changes can be too low. Mean field methods also perform very poorly in the sleep phase, presumably because they make too severe an approximation. Further, it is not easy in the Boltzmann machine to express strong priors as to the nature of the hidden units, *ie* to specify the sort of causes (low dimensional, sparse, factorial, *etc*) that it should extract. Correlations amongst the visible units can be modeled without there being a clear signal in the activities of the hidden units – making it hard to use for representational learning.

Note that the requirement for approximations is not a requirement just for learning. As with other ML density estimation techniques, the recognition distribution itself determines the potential causes of new inputs, and so Gibbs sampling or mean field methods are necessary for the analysis phase too.

11.10 The Helmholtz Machine

The Helmholtz machine is another way of learning representations by performing density estimation. It was explicitly designed to accommodate hierarchical architectures, with complex multi-layer representations. It involves two interacting networks, a top-down one that implements the generative model, and a bottom-up one that implements the recognition model. The Helmholtz machine manipulates these models almost completely symmetrically.

We first see that the simplest version of the Helmholtz machine requires application of the delta rule in two phases of learning, and that it performs the statistical technique of factor analysis (FA). We then describe the nonlinear version of the Helmholtz machine which can capture higher order statistical structure in $P_I[\mathbf{u}]$ than FA, but which nevertheless uses a similar underlying delta rule.

To review, the full generative model for FA is $P[\mathbf{u}, \mathbf{v}; \mathcal{G}]$ specified in equation 11.14 as:

$$P[\mathbf{v}; \mathcal{G}] \sim \mathcal{N}[0, \sigma^2 \mathbb{I}] \tag{11.71}$$

$$P[\mathbf{u}|\mathbf{v}; \mathcal{G}] \sim \mathcal{N}[\mathbf{G}\mathbf{v}, \Psi] \tag{11.72}$$

where the u_b are mutually independent given $\mathbf{v}$, but have different variances $\Psi = \mathrm{diag}(\tau_1^2, \dots, \tau_n^2)$. These different variance components are extra parameters in the generative model, and are what allow FA to ignore a highly variable, but uninformative single input. In this case, it turns out that the recognition distribution $P[\mathbf{v}|\mathbf{u}; \mathcal{G}]$ is Gaussian with mean $\mathbf{W}^*\mathbf{u}$ for a particular $\mathbf{W}^*$ and a covariance matrix Σ^* that does not depend on $\mathbf{u}$. Expressions for these are given in equation 11.17 and 11.18.

Figure 11.11 shows the network for the linear Helmholtz machine. Rather than use these optimal values $\mathbf{W}^*$ and Σ^*, it specifies an approximate distribution

$$\mathcal{Q}[\mathbf{v}; \mathbf{u}, \mathbf{W}, \Sigma] \sim \mathcal{N}[\mathbf{W}\mathbf{u}, \Sigma] \tag{11.73}$$

with parameters the bottom-up weights $\mathbf{W}$ and the recognition covariance matrix Σ. It then adapts these parameters to make

$$\mathcal{Q}[\mathbf{v}; \mathbf{u}, \mathbf{W}, \Sigma] = P[\mathbf{v}|\mathbf{u}; \mathcal{G}] \tag{11.74}$$

We substitute $\mathcal{Q}[\mathbf{v}; \mathbf{u}, \mathbf{W}, \Sigma]$ into the function in equation 11.11

$$\mathcal{F}[\mathbf{u}, \mathcal{Q}; \mathcal{G}] \equiv \mathcal{F}[\mathbf{u}, \mathcal{G}, \Psi, \mathbf{W}, \Sigma] \tag{11.75}$$

to perform the alternating minimisation with respect to $\mathcal{G}$ (and τ^2) and then with respect to $\mathbf{W}, \Sigma$.

The derivatives of the part of $\mathcal{F}[\mathbf{u}, \mathcal{Q}; \mathcal{G}]$ that is dependent on $\mathbf{G}$ is given by:

$$-\frac{\partial}{\partial \mathbf{G}_{ba}} \left\{-\log P[\mathbf{u}, \mathbf{v}; \mathbf{G}, \Psi]\right\} = \frac{1}{\tau_b^2} [\mathbf{u} - \mathbf{G}\mathbf{v}]_b \, v_a, \tag{11.76}$$

$$-\frac{\partial}{\partial \left(\tau_b^2\right)} \left\{-\log P[\mathbf{u}, \mathbf{v}; \mathbf{G}, \Psi]\right\} = \frac{1}{2\tau_b^4} \left([\mathbf{u} - \mathbf{G}\mathbf{v}]_b^2 - \tau_b^2\right). \tag{11.77}$$

These both involve prediction errors, and so lead to delta learning rules for the parameters of the generative model. The derivatives are important

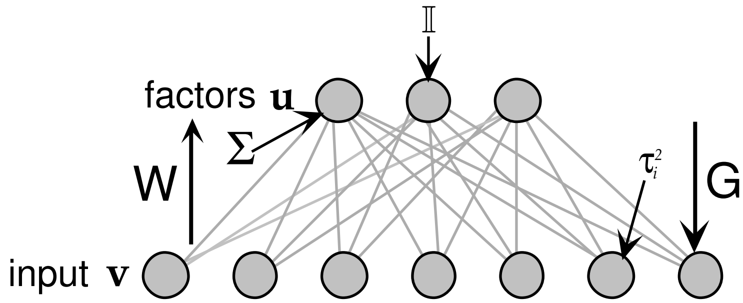

Figure 11.11: The network for factor analysis. The figure shows the linear version of the Helmholtz machine. The top-down variances and generative weights ($\mathbb{I}$ for $\mathbf{v}$; $\mathbf{G}$ and $\Psi = \mathrm{diag}(\tau_1^2, \ldots, \tau_n^2)$ for $\mathbf{u}$) define a simple linear Gaussian generative model for the inputs. Bottom up weights and covariances ($\mathbf{W}$ and Σ) define the inverse to this generative model, which should report $P[\mathbf{v}|\mathbf{u}; \mathcal{G}]$.

if $\mathcal{F}[\mathbf{u}, \mathcal{Q}, \mathcal{G}]$ is minimised in two phases. In the 'wake' phase (the names are inspired by the Boltzmann machine, but the nature of the phases is different), stochastic gradient descent of the full upper bound to the negative log likelihood (in equation 11.19) with respect to the generative model is implemented. This requires sampling $\mathbf{u}$ from $P_I[\mathbf{u}]$, then sampling $\mathbf{v}$ from $\mathcal{N}[\mathbf{W}\mathbf{u}, \Sigma]$ and changing $\mathbf{G}$ and Ψ proportionally to the derivatives in equations 11.76 and 11.77, using these values of $\mathbf{u}$ and $\mathbf{v}$. This implements one of the two alternating minimisation steps that we have been using for density estimation.

The other step requires minimising $\mathcal{F}[\mathbf{u}, \mathcal{Q}, \mathcal{G}]$ with respect to the parameters of the recognition model $\mathcal{Q}[\mathbf{v}; \mathbf{u}, \mathbf{W}, \Sigma]$. Unfortunately, stochastic gradient descent with respect to $\mathbf{W}, \Sigma$ is not so easy, since $\mathcal{Q}[\mathbf{v}; \mathbf{u}, \mathbf{W}, \Sigma]$ appears in two terms of $\mathcal{F}[\mathbf{u}, \mathcal{G}, \Psi, \mathbf{W}, \Sigma]$. However, we know that at the optimal values $\mathbf{W}^*$, Σ^*, $\mathcal{Q}[\mathbf{v}; \mathbf{u}, \mathbf{W}^*, \Sigma^*] = P[\mathbf{v}|\mathbf{u}; \mathcal{G}]$. Therefore, it is sensible to attempt to minimise the reverse measure of the Kullback Leibler divergence between the two distributions:

$$\int_{\mathbf{u}} P[\mathbf{u}] KL\left[P[\mathbf{v}|\mathbf{u}], \mathcal{Q}[\mathbf{v}; \mathbf{u}]\right] d\mathbf{u} = \int_{\mathbf{u}} P[\mathbf{u}] \int_{\mathbf{v}} P[\mathbf{v}|\mathbf{u}] \log \frac{P[\mathbf{v}|\mathbf{u}]}{\mathcal{Q}[\mathbf{v}; \mathbf{u}]} \quad (11.78)$$

$$= \int_{\mathbf{u},\mathbf{v}} P[\mathbf{v}, \mathbf{u}] \log \frac{P[\mathbf{v}|\mathbf{u}]}{\mathcal{Q}[\mathbf{v}; \mathbf{u}]} \quad (11.79)$$

by sampling both $\mathbf{u}$ and $\mathbf{v}$ from the generative model $P[\mathbf{u}, \mathbf{v}]$ in equations 11.71 and 11.72 and then moving down the gradient

$$\frac{\partial}{\partial \mathbf{W}_{ab}} \left\{ -\log \mathcal{Q}[\mathbf{v}; \mathbf{u}] \right\} = \left[\Sigma^{-1} \left(\mathbf{v} - \mathbf{W}\mathbf{u} \right) \right]_a u_b. \quad (11.80)$$

If Σ is diagonal, this is again the delta rule. An equivalent gradient rule can be defined for the diagonal components of Σ too. This is called the 'sleep' phase. The network fantasises complete patterns from its present generative model, and uses them to train the recognition inverse to that model. There are ways of incorporating non-diagonal Σ, but they all have some problems.

The net result is that one can implement an approximation to the statistical technique of factor analysis with delta learning rules, but using two different phases of learning, like the Boltzmann machine. Wake and sleep play different roles for the Helmholtz machine – wake makes the generative model more like the statistical inverse of the recognition model, on patterns $\mathbf{u}$ provided by the world; sleep makes the recognition model more like the statistical inverse of the generative model, on patterns provided by the generative model itself. There is no equivalent of unlearning, as in the sleep phase of the Boltzmann machine. Note that the names for the phases remain metaphorical.

Learning and activation are purely local at each synapse. There is no need for synaptic normalisation, since the task of prediction disciplines the weights from getting infinitely large. Saturation can still be used to prevent the weights from going negative, but, since the algorithm is performing stochastic ascent, this may prevent it from reaching good solutions. Unlike the Boltzmann machine, the Helmholtz machine does not require Gibbs sampling, since it is possible to sample states directly according to the relevant probability distributions in both the recognition and generative models. As a minor point, the weights between the layers are also not necessarily symmetric.

The generative model in factor analysis is only appropriate for inputs that satisfy its simple underlying linear and Gaussian form. However, one underlying heuristic, that of prior independence amongst the factors $\mathbf{v}$ is more generally important. One can build non-linear hierarchical architectures that also embody this principle, which attempt to extract more and more sophisticated correlations amongst inputs in higher and higher layers. The simplest form of non-linear Helmholtz machine uses binary units $\mathbf{u}$ and $\mathbf{v}$, like the Boltzmann machine, and replaces equations 11.71 and 11.72 with:

$$
\begin{aligned}
P[\mathbf{v};\mathcal{G}] &= \prod_a \left[\sigma(g^v_a)\right]^{v_a}\left[1-\sigma(g^v_a)\right]^{1-v_a} && (11.81)\\
P[\mathbf{u}|\mathbf{v};\mathcal{G}] &= \prod_b \left[\sigma\left([\mathbf{G}\mathbf{v}]_b\right)\right]^{u_b}\left[1-\sigma\left([\mathbf{G}\mathbf{v}]_b\right)\right]^{1-u_b}. && (11.82)
\end{aligned}
$$

where $\mathbf{g}^v$ are generative bias weights for $\mathbf{v}$ which specify how likely they are to be active. These equations preserve the unconditional independence of the $\{v_a\}$ and the conditional independence of $\mathbf{u}$ given the value of $\mathbf{v}$. However, they use the non-linear sigmoid function $\sigma(\cdot)$ from the Boltzmann machine (equation 11.69). Activations during the wake and sleep phases are just as for the linear Helmholtz machine, except sampling from binomial distributions with given means rather than Gaussian distributions with given means and covariances. Further, since

$$
-\frac{\partial}{\partial \mathbf{G}_{ba}}\left\{-\log P[\mathbf{u},\mathbf{v};\mathbf{G}]\right\} = \left[u_b - \sigma\left([\mathbf{G}\mathbf{v}]_b\right)\right] v_a, \qquad (11.83)
$$

the delta rule is still the appropriate way to change the weights, except that the linear mean activation $\mathbf{Gv}$ of equation 11.76 is replaced by the nonlinear mean activation $\sigma\left([\mathbf{Gv}]_b\right)$. Adaptation is based on the reconstruction error between the actual activity u_b and the activity predicted using the top-down connections.

A natural model for the recognition distribution has a similar form to equation 11.82 using bottom-up weights $\mathbf{W}$:

$$Q[\mathbf{v};\mathbf{u}] = \prod_a \left[\sigma\left([\mathbf{Wu}]_a\right)\right]^{v_a} \left[1-\sigma\left([\mathbf{Wu}]_a\right)\right]^{1-v_a}. \qquad (11.84)$$

with v_a and $v_{a'}$ being conditionally independent given $\mathbf{u}$. This is only an approximation, since the $\mathbf{v}$ are really made conditionally *dependent* because of the way that they interact to generate $\mathbf{u}$ in equation 11.82.

Just as for FA, it is much easier to minimise $\mathbf{W}$ with respect to the Kullback-Leibler divergence the incorrect way round. In this case, again, samples $\mathbf{v}$ and then $\mathbf{u}$ are generated from the generative model. Since

$$-\frac{\partial}{\partial \mathbf{W}_{ab}}\left\{-\log Q[\mathbf{v};\mathbf{u}]\right\} = \left[v_a - \sigma\left([\mathbf{Wu}]_a\right)\right] u_b \qquad (11.85)$$

the recognition weights can again be adapted using the delta rule. Adaptation here is based on a similar (but bottom-up) reconstruction error, except that the samples come from the network's fantasies rather than true inputs from the environment.

The particular advantage of this non-linear Helmholtz machine is that one can have multiple hierarchical layers of causes $\mathbf{v}$. Activation and learning remain the same, with stochastic choices being made for the activities within each layer bottom-up during wake and top-down during sleep. By contrast, it does not make much sense to perform FA in multiple layers, since it cannot generate more complicated distributions with many layers than with one. It is easy to have multiple layers in the Boltzmann machine, since all the units are interconnected.

The non-linear Helmholtz machine uses sampling, an approximate recognition distribution, and also the incorrect Kullback-Leibler divergence during the sleep phase. Sampling is benign compared with the use of Gibbs sampling in the Boltzmann machine, since samples can be generated in a feedforward manner without having to reach equilibrium of a Markov chain. Also, learning does not depend on the difference between two noisy quantities. The approximation in the recognition distribution that the activities $\{v_a\}$ are conditionally independent given the input $\mathbf{u}$ is quite severe – and there are many interesting psychophysical phenomena such as binocular rivalry and alternating percepts that show that this is unlikely to be adequate. Weakening this constraint is a major topic of investigation, as is using recognition methods such as mean field that avoid it altogether. Mean field methods also avoid the incorrect Kullback-Leibler divergence, which has been shown to be problematical in various cases.

Various other versions of hierarchically layered belief networks are also being actively explored as ways of performing representational learning. The degrees of freedom in designing a model include the form of the architecture, the statistical nature of the units (Gaussian, Bernoulli, *etc*), the parameterisation of the degrees of freedom, and the use of exact, approximate (*eg* mean field), or sampling (*eg* Gibbs sampling) methods for inference and learning.

11.11 Appendices

A) Summary of the Generative and Recognition Models

Model	Generative Distribution	Recognition Distribution
FA	$P[\mathbf{v};\mathcal{G}] \sim \mathcal{N}[\mathbf{0},\sigma^2\mathbb{I}]$ $P[\mathbf{u}\vert\mathbf{v};\mathcal{G}] \sim \mathcal{N}[\mathbf{G}\mathbf{v},\Psi]$ $\Psi = \mathrm{diag}\left(\tau_1^2,\tau_2^2,\ldots,\tau_n^2\right)$	$P[\mathbf{v}\vert\mathbf{u};\mathcal{G}] \sim \mathcal{N}[\Sigma^*\mathbf{G}^T\Psi^{-1},\Sigma^*]$ $\Sigma^* = \left(\frac{\mathbb{I}}{\sigma^2}+\mathbf{G}^T\Psi^{-1}\mathbf{G}\right)^{-1}$
mixture of Gaussians	$P[v_a\!=\!1] = \pi_a$ $P[\mathbf{u}\vert\mathbf{v};\mathcal{G}] \sim \mathcal{N}[\mathbf{g}^a,\mathbb{I}]$	$P[v_a\!=\!1\vert\mathbf{u};\mathcal{G}] \propto \pi_a e^{-\lVert\mathbf{u}-\mathbf{g}^a\rVert^2/2}$
sparse coding	$P[\mathbf{v};\mathcal{G}] = \prod_a e^{-f(v_a)}$ $P[\mathbf{u}\vert\mathbf{v};\mathcal{G}] \sim \mathcal{N}[\mathbf{G}\mathbf{v},\tau^2\mathbb{I}]$	$Q[\mathbf{v};\mathbf{u}] = \delta(\mathbf{v}-\hat{\mathbf{v}})$
ICA	$P[\mathbf{v};\mathcal{G}] = \prod_a e^{-f_a(v_a)}$ $P[\mathbf{u}\vert\mathbf{v};\mathcal{G}] = \delta(\mathbf{u}-\mathbf{G}\mathbf{v})$	$P[\mathbf{v}\vert\mathbf{u};\mathbf{G}] = \vert\mathbf{G}\vert\delta(\mathbf{u}-\mathbf{G}\mathbf{v})$
Boltzmann machine	$P[\mathbf{u},\mathbf{v}] \propto e^{-\sum_{i<j}\mathbf{G}_{ij}z_iz_j}$	$P[\mathbf{v}\vert\mathbf{u}]$ for sampling $Q[\mathbf{v};\mathbf{u}]\!=\!\prod_a q_a^{v_a}(1-q_a)^{(1-v_a)}$ for mean field
Helmholtz machine	$P[\mathbf{v};\mathbf{G}] = \prod_a\left[\sigma(g_a^v)\right]^{v_a}$ $\times\left[1-\sigma(g_a^v)\right]^{1-v_a}$ $P[\mathbf{u}\vert\mathbf{v};\mathbf{G}] = \prod_b\left[\sigma\left([\mathbf{G}\mathbf{v}]_b\right)\right]^{u_b}$ $\times\left[1-\sigma\left([\mathbf{G}\mathbf{v}]_b\right)\right]^{1-u_b}$	$Q[\mathbf{v};\mathbf{u}] = \prod_a\left[\sigma\left([\mathbf{W}\mathbf{u}]_a\right)\right]^{v_a}$ $\times\left[1-\sigma\left([\mathbf{W}\mathbf{v}]_a\right)\right]^{1-v_a}$

All these models for ML density estimation are discussed in detail in the text, although the forms quoted are just for the simplest cases. For the recognition models, $P[\mathbf{v}|\mathbf{u};\mathbf{G}]$ indicates the true posterior recognition over $\mathbf{v}$ given $\mathbf{u}$; $Q[\mathbf{v};\mathbf{u}]$ indicates an approximate recognition distribution whose parameters are adjusted either off-line (for the Helmholtz machine) or on-line (the other models) to minimise $\mathcal{F}[\mathbf{u},Q;\mathcal{G}]$. $\mathcal{N}[\mathbf{a},\Omega]$ is a multivariate Gaussian distribution with mean $\mathbf{a}$ and covariance matrix Ω, $\sigma(c) = 1/(1+\exp(-c))$, $\delta(\mathbf{v}-\tilde{\mathbf{v}})$ is a delta-function distribution, and $|\mathbf{G}|$ is the absolute value of the determinant of the invertible matrix $\mathbf{G}$.

B) Gibbs Sampling for the Boltzmann Machine

The Boltzmann machine generates samples using a stochastic simulation or Markov chain Monte-Carlo technique called Gibbs sampling. We describe this simulation technique in the context of the complete distribution $P[\mathbf{z};\mathbf{G}]$, where $\mathbf{z}$ is a shorthand notation for both $\mathbf{u}$ and $\mathbf{v}$, since we need this as well as the conditional distribution $P[\mathbf{v}|\mathbf{u};\mathbf{G}]$, and the operations are essentially the same.

A Markov chain is a probabilistic system whose state ($\mathbf{z}(t)$ here at time t) evolves over time in a simple stochastic way. The update to $\mathbf{z}(t+1)$ is made memoryless, in that it depends only on the current state $\mathbf{z}(t)$:

$$P_T[\mathbf{z}(t+1)|\mathbf{z}(1),\ldots,\mathbf{z}(t)] = P_T[\mathbf{z}(t+1)|\mathbf{z}(t)] \tag{11.86}$$

so that the history of how the state came to be $\mathbf{z}(t)$ is unimportant for determining $\mathbf{z}(t+1)$. If there are only finitely many possible values of $\mathbf{z}$, then the collection of update probabilities is called the transition matrix of the finite state Markov chain.

Many Markov chains have an equilibrium distribution – *ie* a distribution $\pi(\mathbf{z})$ over the states such that no matter what state the Markov chain is started in, it will ultimately tend to occupy states with probabilities $\pi(\mathbf{z})$. That is

$$\lim_{t\to\infty} P[\mathbf{z}(t)=\mathbf{z}] = \pi(\mathbf{z}). \tag{11.87}$$

If we could arrange for a Markov chain to have as its equilibrium distribution $P[\mathbf{z};\mathbf{G}]$, then we could try to get *samples* from $P[\mathbf{z};\mathbf{G}]$ by starting the chain from any state, letting it evolve for a long time τ, and then recording $\mathbf{z}(\tau), \mathbf{z}(\tau+1), \ldots$. Such samples are useful for calculating the averages that are required for learning.

The most important condition that the transition probabilities $P_T[\mathbf{z}'|\mathbf{z}]$ of a Markov chain must satisfy to have equilibrium distribution $P[\mathbf{z};\mathbf{G}]$ is invariance to updates:

$$P[\mathbf{z}';\mathbf{G}] = \sum_{\mathbf{z}} P[\mathbf{z};\mathbf{G}]P_T[\mathbf{z}'|\mathbf{z}] \tag{11.88}$$

and the form of stochastic simulation used in the Boltzmann machine specifies a $P_T[\mathbf{z}'|\mathbf{z}]$ for which this is true. It arranges for the appropriate equilibrium distribution by designing transition probabilities that satisfy a condition called *detailed balance.* This means that

$$P[\mathbf{z};\mathbf{G}]P_T[\mathbf{z}'|\mathbf{z}] = P[\mathbf{z}';\mathbf{G}]P_T[\mathbf{z}|\mathbf{z}']. \tag{11.89}$$

Detailed balance is a sufficient (but not a necessary) condition for $P[\mathbf{z};\mathbf{G}]$ to be the equilibrium distribution. Note that $P[\mathbf{z};\mathbf{G}]$ only occurs in the detailed balance condition in *ratios.* This obviates any requirement to calculate the partition functions in equation 11.59 or 11.65.

There is one last complication in the Boltzmann machine in arranging for this, namely the existence of multiple units. It is unreasonable to impose the requirement that information should propagate in a global manner around the network in changing the state from $\mathbf{z}$ to $\mathbf{z}'$. Gibbs sampling instead defines a whole sequence of operators, $P_{T^i}[\mathbf{z}'|\mathbf{z}]$, one for each unit i. Each of these specifies the probability of a change to the state of just its own unit; nevertheless, one can show that if they are applied asynchronously (*ie* one at a time), either in sequence or randomly, then the equilibrium distribution will still be $P[\mathbf{z}]$. The transition matrix $P_{T^i}[\mathbf{z}'|\mathbf{z}]$ sets z'_i to be 0 or 1 conditional on the states of all the other units, in a way that satisfies equation 11.89 trivially. Take just unit z_1 and write $\mathbf{z}_{\bar{1}}$ for the states of the other units. Gibbs sampling ignores the previous state of z_1 and sets

$$\frac{P_{T^1}[z_1 = 1, \mathbf{z}_{\bar{1}}|\mathbf{z}]}{P_{T^1}[z_1 = 0, \mathbf{z}_{\bar{1}}|\mathbf{z}]} = \frac{P[z_1 = 1|\mathbf{z}_{\bar{1}}; \mathbf{G}]}{P[z_1 = 0|\mathbf{z}_{\bar{1}}; \mathbf{G}]} = e^{-\beta \sum_{j>1} \mathbf{G}_{1j} z_j} \tag{11.90}$$

where the last form comes directly from the energy function $E(\mathbf{z})$ (note that $\mathbf{G}_{11} = 0$). Since

$$P_{T^1}[z_1(t+1) = 1|\mathbf{z}_{\bar{1}}(t)] + P_{T^1}[z_1(t+1) = 0|\mathbf{z}_{\bar{1}}(t)] = 1, \tag{11.91}$$

we can solve to get the simple term

$$P_{T^1}[z_1(t+1) = 1|\mathbf{z}_{\bar{1}}(t)] = \sigma\left(\beta \sum_{j>1} \mathbf{G}_{1j} z_j(t)\right) \tag{11.92}$$

where

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{11.93}$$

is the conventional sigmoid function. Also $\mathbf{z}_{\bar{1}}(t+1) = \mathbf{z}_{\bar{1}}(t)$, since, by definition, only the state of unit 1 changes under $P_{T^1}[\mathbf{z}'|\mathbf{z}]$. Remember that the weight matrix $\mathbf{G}$ is symmetric (so that E is an energy function).

To recap, Gibbs sampling is a way of arranging for the states $\mathbf{z}(t)$ to be selected according to a particular probability distribution $P[\mathbf{z}]$. The net requirement involves visiting each node individually and setting its state to 0 or 1 stochastically according to a probability distribution determined as the sigmoid of its net input from the other units. This is a stochastic form of the standard update rule for units in the Hopfield net. It is a purely local way of exploring what amounts to a complicated distribution with hidden units in which the probabilities are non-linear in the parameters. Unfortunately, it suffers from being extremely slow.

Sampling of $P[\mathbf{z}; \mathbf{G}]$ is called the *negative* or sleep phase of learning. Since $P[\mathbf{z}; \mathbf{G}]$ is only the *stationary* or terminal distribution of the Markov chain, it is usually necessary to perform a number of Gibbs sampling state-updates before extracting samples $\mathbf{v}^\nu$. The early phase of Gibbs sampling before the patterns are used is called settling. Also, it can be advantageous to use *annealing, ie* starting off with a small value of β as samples are taken, and increasing it slowly towards 1. When β is small, the Markov chain is rapidly mixing, in the sense that there are no great energy barriers between states. As β rises, the effect of small energy differences between states gets larger and so the Markov chain takes substantially longer to explore state space. Choosing good annealing schedules for arbitrary chains is rather hard.

Sampling from the recognition distribution $P[\mathbf{v}|\mathbf{u};\mathcal{G}]$ for a particular input $\mathbf{u}$ involves similar operations. One just forces the values of the inputs to stick at $\mathbf{u}$ (this is called *clamping*), and then only updates the representational units $\mathbf{v}$ in the network. This works since the only difference between $P[\mathbf{v}|\mathbf{u};\mathcal{G}]$ and $P[\mathbf{v},\mathbf{u};\mathcal{G}]$ is the set of values of $\mathbf{z}$ that are summed to form the partition function, and, by construction, the partition function does not appear in the single-unit transition term in equation 11.90. Sampling of $P[\mathbf{v}|\mathbf{u};\mathcal{G}]$ is called the *positive* or wake phase of learning.

C) Mean-Field Methods for the Boltzmann Machine

An alternative to using Gibbs sampling to estimate the averages in equations 11.66 and 11.67 is to employ parameterised approximating distributions $Q[\mathbf{v}|\mathbf{u}]$ and $Q[\mathbf{z}']$, and to choose parameters that optimise the fit to the desired distributions $P[\mathbf{v}|\mathbf{u};\mathbf{G}]$ and $P[\mathbf{z}';\mathbf{G}]$. The approximating distributions should be flexible enough to capture enough of the structure in these distributions, but simple enough that they are easy to manipulate mathematically.

The simplest mean field distribution (and the reason for its name) treats all the units as being mutually independent. Consider the task of approximating $P[\mathbf{z};\mathbf{G}]$ by

$$Q[\mathbf{z}] = \prod_i q_i^{z_i}(1-q_i)^{(1-z_i)} \tag{11.94}$$

This distribution gives each unit i a mean field parameter q_i which is its mean activity.

As usual in this chapter, the discrepancy between $Q[\mathbf{z}]$ and $P[\mathbf{z};\mathbf{G}]$ is measured by the Kullback-Leibler divergence:

$$\begin{aligned} KL[Q[\mathbf{z}],P[\mathbf{z};\mathbf{G}]] &= -\textstyle\sum_{\mathbf{z}} Q[\mathbf{z}]\log P[\mathbf{z};\mathbf{G}] + \sum_{\mathbf{z}} Q[\mathbf{z}]\log Q[\mathbf{z}] \\ &= -\tfrac{1}{2}\textstyle\sum_{i\neq j}\mathbf{G}_{ij}q_iq_j + \sum_i q_i\log q_i + (1-q_i)\log(1-q_i) \end{aligned} \tag{11.95}$$

We can choose the parameters $\{q_i\}$ to minimise this Kullback-Leibler divergence. Differentiating with respect to q_i, we get

$$\nabla_{q_i} KL[Q[\mathbf{z}],P[\mathbf{z};\mathbf{G}]] = -\left[\mathbf{G}\mathbf{q}\right]_i + \log\frac{q_i}{1-q_i} \tag{11.96}$$

remembering that $\mathbf{G}_{ii} = 0$. We can find the minimum with respect to q_i by setting this to 0, at

$$q_i^* = \sigma\left(\left[\mathbf{G}\mathbf{q}\right]_i\right). \tag{11.97}$$

We can therefore find the mean field optimum by visiting units either systematically or at random and changing their state according to equations such as equation 11.97. This implements coordinate-wise descent of the Kullback-Leibler divergence, and therefore the quality of the approximation improves monotonically with successive iterations. Note that this is just the same update as for Gibbs sampling (equation 11.92), except that it is a deterministic update to find a parameter giving the probability that a Bernoulli variable takes the value 1, rather than being the probability of setting a Bernoulli variable to 1 in a sampling scheme.

Given the terminal mean field activities $\mathbf{q}^*$, an approximation to the average in equation 11.67 is easy to compute using

$$\langle z_i' z_j' \rangle_- = \sum_{\mathbf{z}} P[\mathbf{z}; \mathbf{G}] z_i z_j \sim \sum_{\mathbf{z}} Q[\mathbf{z}] z_i z_j = q_i^* q_j^*. \tag{11.98}$$

A mean field approximation to the conditional distribution can be similarly derived, by fixing the states of the $\mathbf{u}$ units to the values in the input and only parameterising the activities of the $\mathbf{v}$ units.

Mean field distributions are an important alternative to sampling methods for estimating or approximating complex distributions. A coterie of sophisticated methods inspired by them has been developed for the class of networks that includes the Helmholtz machine, and offers an important alternative to the stochastic wake-sleep algorithm that we described.

Mean field methods have the advantage that they are deterministic, and therefore do not depend on the vagaries of sampling, and that in some circumstances, as the number of units increases, they become ever *more* accurate. They can also provide strict *bounds* which is useful to ensure that learning proceeds correctly. They have the disadvantage that they are hard to refine in a natural way, and, in some cases, are also substantially more complicated, making questionable their biological implementation.

11.12 Further Reading and References

Further Reading

Barlow, HB (1989) Unsupervised learning. *Neural Computation* **1**:295-311.

Becker, S & Plumbley, M (1996) Unsupervised neural network learning procedures for feature extraction and classification. *International Journal of Applied Intelligence* **6**:185-203.

Bishop, CM (1995) *Neural Networks for Pattern Recognition*. Oxford:Clarendon Press, Oxford University Press.

Carpenter, GA & Grossberg, S, editors (1991) *Pattern Recognition by Self-Organizing Neural Network*. Cambridge, MA:MIT Press.

Hinton, GE (1989) Connectionist learning procedures. *Artificial Intelligence* **40**:185-234.

Mallat, SG (1998) *A Wavelet Tour of Signal Processing.* San Diego: Academic Press.

References

Baddeley, R, Abbott, LF, Booth, MCA, Sengpiel, F, Freeman, T, Wakeman, EA & Rolls, ET (1997) Responses of neurons in primary and inferior temporal visual cortices to natural scenes. *Proceedings of the Royal Society of London Series B* **264**:1775-1783.

Barlow, HB (1961) Possible principles underlying the transformation of sensory messages. In WA Rosenblith, editor, *Sensory Communication*. Cambridge, MA:MIT Press.

Becker, S & Hinton, GE (1992) A self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature* **355**:161-163.

Bell AJ & Sejnowski TJ (1995) An information maximisation approach to blind separation and blind deconvolution. *Neural Computation* **7**:1129-1159.

Bell AJ & Sejnowski TJ (1996) Learning the higher-order structure of a natural sound. *Network* **7**:261-267.

Dayan, P, Hinton, GE, Neal, RM & Zemel, RS (1995) The Helmholtz machine. *Neural Computation* **7**:889-904.

Dempster, AP, Laird, NM & Rubin, DB (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B* **39**:1-38.

Grenander, U (1976-1981) *Lectures in Pattern Theory I, II and III: Pattern Analysis, Pattern Synthesis and Regular Structures*. Berlin:Springer-Verlag.

Heeger, DJ (1992). Normalization of cell responses in cat striate cortex. *Visual Neuroscience* **9**:181-197.

Hinton, GE, Dayan, P, Frey, BJ & Neal, RM (1995) The wake-sleep algorithm for unsupervised neural networks. *Science* **268**:1158-1160.

Hinton, GE & Sejnowski, TJ (1986) Learning and relearning in Boltzmann machines. In DE Rumelhart, JL McClelland & the PDP research group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. Cambridge, MA MIT Press, 282-317.

Jordan, MI, Ghahramani, Z, Jaakkola, TS & Saul, LK (1988) An introduction to variational methods for graphical models. In MI Jordan, editor, *Learning in Graphical Models.* Dordrecht: Kluwer: 163-174.

Kawato, M, Hayakama, H & Inui, T (1993) A forward-inverse optics model of reciprocal connections between visual cortical areas. *Network: Computation in Neural Systems* **4**:415-422.

Karasaridis, A & Simoncelli, EP (1996) A filter design technique for steerable pyramid image transforms. *International Conference on Acoustics, Speech and Signal Processing.* New York, NY: IEEE, 2387-2390.

Li, Z & Atick, JJ (1994). Toward a theory of striate cortex. *Neural Computation* **6**:127-146.

MacKay, DJC (1996) *Maximum Likelihood and Covariant Algorithms for Independent Components Analysis* Unpublished manuscript.

MacKay, DM (1956) The epistemological problem for automata. In CE Shannon & J McCarthy, editors, *Automata Studies*. Princeton, NJ:Princeton University Press, 235-251.

Neal, RM & Dayan, P (1997) Factor Analysis using delta-rule wake-sleep learning. *Neural Computation* **9**:1781-1803.

Neal, RM & Hinton, GE (1994) A view of the EM algorithm that justifies incremental, sparse, and other variants. In MI Jordan, editor, *Learning in Graphical Models.* Dordrecht: Kluwer:355-368.

Nowlan, SJ (1991) *Soft Competitive Adaptation: Neural Network Learning Algorithms Based on Fitting Statistical Mixtures*. PhD Thesis, Department of Computer Science, Carnegie-Mellon University.

Olshausen, BA & Field, DJ (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* **381**:607-609.

Olshausen, BA & Field, DJ (1997) Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research* **37**:3311-3325.

Pearlmutter, BA & Parra, LC (1996) A context-sensitive generalization of ICA. In S-I Amari, L Xu, L-W Chan & I King, editors, *Proceedings ICONIP '96*. Singapore:Springer-Verlag, 151-157.

Rao, RPN & Ballard, DH (1997) Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural Computation* **9**:721-763.

Rissanen, J (1989) *Stochastic Complexity in Statistical Inquiry*. Singapore:World Scientific.

Rubin, DB & Thayer, DT (1982) EM algorithms for ML factor analysis. *Psychometrika* **47**:69-76.

Schmidhuber, J (1992) Learning factorial codes by predictability minimization. *Neural Computation* **4**:863-879.

Shapiro, JM (1993). Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing* **41**: 3445-3462.

Simoncelli, EP & Adelson, EH (1990) Subband image coding. In JW Woods, editor, *Subband Transforms* Norwell, MA: Kluwer, 143-192.

Simoncelli, EP (1997) Statistical models for images: Compression, restoration and synthesis. *31st Asilomar Conference on Signals, Systems and Computers.* Pacific Grove, CA: IEEE.

Simoncelli, EP, Freeman, WT, Adelson, EH & Heeger, DJ (1992) Shiftable multiscale transforms. *IEEE Transactions on Information Theory,* **38**:587-607.

Simoncelli, EP & Freeman, WT (1995) The steerable pyramid: A flexible architecture for derivative computation. *International Conference on Image Processing.* Los Alamitos, CA: IEEE Computer Society Press, 444-447.

Simoncelli, EP & Schwartz, O (1999) Image statistics and cortical normalization models. In M Kearns, S Solla & D Cohn, editors, *Advances in Neural Information Processing, 11.* Cambridge, MA: MIT Press.

Tipping, ME & Bishop, CM (1997) *Probabilistic Principal Component Analysis* NCRG/97/010, Aston University, Birmingham, England.

Titterington, DM, Smith, AFM & Makov, UE (1985) *Statistical Analysis of Finite Mixture Distributions*. Chichester, NY:Wiley.

Zemel, RS (1994) *A Minimum Description Length Framework for Unsupervised Learning*. PhD Dissertation, Computer Science, University of Toronto.

11.13 Exercises

The exercises probe the consequences of using various methods discussed in the chapter to learn how to represent patterns whose true structure is known. The patterns consist of binary or real versions of horizontal and/or vertical bars in an $n \times n$ grid. Four basic pattern sets are:

B1 Each of the n horizontal bars is turned on independently with probability p. If a bar is on, then the pixels of the input it covers are turned on (independently) with probability 0.9; otherwise the pixels of the input are turned on with probability 0.1.

B2 The same as B1 except that with probability 0.5, horizontal bars are generated, and with probability 0.5, vertical bars are generated.

R1 The strength of each of the n horizontal bars is chosen independently from a Gaussian distribution with mean 0 and variance 1. The strength of each pixel in the input is the sum of the strength of the bar that contains it and an independent Gaussian random variable with mean 0 and variance 0.1.

R2 The same as R1 except that with probability 0.5, horizontal bars are generated, and with probability 0.5, vertical bars are generated.

The structure in each of these is determined by the image generation process.

1. Generate 1000 patterns of the form of R1 and calculate their covariance matrix. What do the structure of the eigenvalues and eigenvectors reveal about the underlying causes for the input patterns. Do the eigenvectors consist of separate horizontal bars? Why is this outcome expected?

2. Repeat exercise 1 for patterns R2 and comment on the changes brought about by the addition of the vertical bars.

3. Repeat exercise 1 in the case that the added independent Gaussian noise has substantially different variances for each pixel. How does this perturb the eigenvectors?

4. Use equations 11.17, 11.18, 11.21 and 11.22 to implement factor analysis on patterns R1 as in exercises 1 and 3. Is there any detectable difference between factor analysis and principal components analysis in this case?

5. Use equations 11.30, 11.32 and 11.33 to implement a mixture of Gaussians model and apply it to R1 and B1. Comment on the representation extracted. As $p \to 0$ for B1, does the representation improve?

6. Generate a set of patterns according to R1 but using a Cauchy distribution with various dispersions for the strengths of the horizontal bars. Use the factor analysis algorithm of exercise 4 to extract the causes – are they affected by the change in distribution? Implement the sparse coding algorithm and ICA to extract the causes instead – are these any more proficient?

7. For $n = 3$, calculate the true probability of each of the 512 patterns in B1 and B2 and the entropy of the whole distribution.

8. Implement the wake-sleep algorithm of equations 11.83 and 11.85 on the patterns in B1 and B2 using variable numbers of hidden units. Evaluate the Kullback-Leibler divergence between the true distribution of the inputs and the distribution created by the generative model – does wake-sleep tend to reduce this? Why might it be difficult to separate the horizontal and vertical bars in B2?