An Information-theoretic Unsupervised Learning Algorithm for Neural Networks

by

Suzanna Becker

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

In the unsupervised learning paradigm, a network of neuron-like units is presented an ensemble of input patterns from a structured environment, such as the visual world, and learns to represent the regularities in that input. The major goal in developing unsupervised learning algorithms is to find objective functions that characterize the quality of the network's representation without explicitly specifying the desired outputs of any of the units. Previous approaches in unsupervised learning, such as clustering, principal components analysis, and information-transmission-based methods, make minimal assumptions about the kind of structure in the environment, and they are good for preprocessing raw signal input. These methods try to model *all* of the structure in the environment in a single processing stage. The approach taken in this thesis is novel, in that our unsupervised learning algorithms do not try to preserve all of the information in the signal. Rather, we start by making strongly constraining assumptions about the kind of structure of interest in the environment. We then proceed to design learning algorithms which will discover precisely that structure. By constraining what kind of structure will be extracted by the network, we can force the network to discover higher level, more abstract features. Additionally, the constraining assumptions we make can provide a way of decomposing difficult learning problems into multiple simpler feature extraction stages. We propose a class of information-theoretic learning algorithms which cause a network to become tuned to spatially coherent features of visual images. Under Gaussian assumptions about the spatially coherent features in the environment, we have shown that this method works well for learning depth from random dot stereograms of curved surfaces. Using mixture models of coherence, these algorithms can be extended to deal with discontinuities, and to form multiple models of the regularities in the environment. Our simulations demonstrate the general utility of the Imax algorithms in discovering interesting, non-trivial structure (disparity and depth discontinuities) in artificial stereo images. This is the first attempt we know of to model perceptual learning beyond the earliest stages of low-level feature extraction, and to model multiple stages of unsupervised learning.

Acknowledgements

First and foremost, I would like to thank my thesis advisor, Geoffrey Hinton, who has illustrated to me what it means to do brilliant research, and equally importantly, how to explain it to others. I do hope these qualities are at least a little bit contagious. I want to thank Geoff for sharing so much of his time, his unparalleled enthusiasm and excitement for research, his endless wealth of ideas and insights, his wonderfully biased opinions, and his unfailing sense of humour.

I thank the members of my thesis committee, Allan Jepson, Demetri Terzopoulos, Ken Sevcik, Rudi Mathon, Rob Tibshirani and Andy Barto, who in many ways have shaped this dissertation, and helped me to see the issues in a broader context. I also thank Yann le Cun, Ray Watrous, Barak Pearlmutter, John Bridle, and Niels da Vitoria Lobo for the valuable discussions we've had. And for their numerous helpful comments on drafts of large portions of this dissertation, I thank Peter Foldiak, Peter Dayan, David Mackay, Greg Dudek, and Richard Mann. Thanks to all the members of the connectionist research group at the University of Toronto for providing a work environment that was both congenial and stimulating, and which I shall miss very much. Rich Zemel and Steve Nowlan have been particularly invaluable as both colleagues and friends, and I thank them for all the discussions and their feedback on every stage of my work. Many thanks to Tony Plate and Drew van Camp for Xerion, and to Carol Plathan and Maureen Smith for sheltering us from UofT's formidable bureaucracy.

In addition to those who have contributed in a direct way to the content of this thesis, there are many people I would like to thank for providing the context in which it was made possible. I thank Bart Selman and Niels da Vitoria Lobo for encouraging me to pursue a PhD in the first place, and for always believing in me. Also, I thank Toni Pitassi, a shining star, whose support has made all the difference, and who has enriched my life a whole lot, and the many other wonderful friends I got to know along the way, who helped me to survive the process, and to find much joy in it too: Marlene Behrmann, Naomi Nishimura, Janet Wiles, Barb Moore, Armin Haken, Arvind Gupta, Evan Steeg, Dave Plaut, Eyal Shavit, Karen Rudi, Diane Horton, and all the "femphdals". And for their love and support and encouragement, I thank my Mom and my Dad. And last but most of all, for providing all of the above, and way more, I thank Steve Garrett.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Connectionist networks

A major goal of artificial intelligence (AI) research is to develop computational models which exhibit performance comparable to that of humans on the every-day information-processing tasks we engage in: extracting meaning from complex patterns of stimuli such as the structure of objects in the visual world, words from acoustic signals, or sentence meaning from text. Human performance on these tasks is characterized by a robustness to noise and ambiguity, and a remarkable efficiency in situations which appear to require the simultaneous satisfaction of many constraints.

A dichotomy in AI research exists between symbolic approaches and connectionist models. They differ most fundamentally on the question of what is considered an appropriate level of explanation for cognitive processes. Advocates of the symbolic approach (e.g. Pylyshyn, 1981; Newell and Simon, 1981) argue that the most appropriate level at which to describe human intelligence is in terms of symbols and symbol-manipulating rules. In contrast, the building blocks for connectionists are model neurons or neural populations; intelligence is manifested in the computation performed by large highly interconnected networks of these elements. While the symbolic approach has been more successful at modeling abstract problem-solving, the connectionist approach has been more fruitful in capturing "lower level" cognitive and perceptual capabilities such as visual and auditory pattern recognition, learning and memory. The reasons connectionist models have been so successful include the following:

1. Parallelism and local control

   Neural network models typically adhere to the principle of strict locality of computation. That is, the computation performed at each network node at any given time step is independent of the processing at other nodes, and depends only on the incoming signals to that node. This results in a very simple, local control mechanism, and permits efficient parallel implementations. Many visual spatial computations (such as are thought to be performed in the retina and early visual cortical areas) involve the repeated application of a simple, spatially localized filter to many image locations in parallel, and can very naturally be mapped onto this sort of model.

   In addition to the obvious advantage of processing efficiency afforded by a parallel computing device, this type of model allows system states to be represented in an economical, simple manner. A system state is simply represented as the pattern of simultaneous activation of the networks' units, so there is no need for extra machinery to maintain state information.

2. Distributed representations

   A concept can be represented by a pattern of activation distributed over a set of units. Therefore, many similar concepts can be efficiently represented in the same network by the activation of overlapping sets of units. Distributed representations allow robust pattern matching based on partial matches. Further, in networks with feedback, a partial representation of a pattern can cause the complete pattern to be filled in. In signal processing tasks such as visual or auditory pattern recognition, robust partial matching and pattern completion are particularly critical. Finally, the properties of distributed memories mentioned here result in a robustness to network damage, and graceful degradation under increasing damage.

3. Adaptation

   The long term memory of a network is represented by the strengths of the connections between units. Learning involves changing these connection strengths. This simple assumption, based on the analogy with biological neural networks, imparts the network with the potential to develop representations appropriate to particular tasks, to adapt to changing environments, and to fine-tune its internal parameters automatically. More traditional AI approaches, including both symbolic AI and computational vision, have had difficulty in situations which require reasoning in the face of uncertainty; in particular, there is the pervasive problem of how to select numerical values involved in combining various sources of evidence, determining thresholds for feature detection, etc. Thus, the capacity to learn these numerical values is a tremendous advantage. Finally, there are situations where no analytical solution to a problem is known, although the problem is well defined in terms of a specification of the inputs and desired outputs of the system (e.g. in signal prediction problems), and an adaptive solution is the only alternative.

Many connectionist models of learning have been proposed. Two major classes of models are supervised (i.e., learning from an external teacher) and unsupervised learning (i.e., discovering some sort of structure in an unlabelled pattern set). This thesis focuses on *unsupervised* learning algorithms for neural networks. We particularly focus on the nature of the *representations* discovered by unsupervised learning methods, and their appropriateness as input for further stages of learning/perceptual processing. The major problem we wish to address is this: how can a system, after experience with a particular environment, learn a representation of the structure in that environment which is useful for subsequent intelligent decision-making, in tasks such as object recognition, exploration, and taking context-specific actions?

In the remainder of this chapter, we define some notation and terminology that will be used in the thesis, discuss the supervised-unsupervised learning distinction, and then outline the contents of the remaining chapters.

1.2 Notational preliminaries

In this section, we introduce some terminology for describing artificial neural networks, and some of the mathematical tools we will use later.

1.2.1 Neural network models

A network consists of a set of parallel processing elements we shall refer to as *units* (model neurons), which are connected by directed *links*. Each unit receives input from other units along zero or more incoming links, computes an *output* or *activation* value, and transmits this signal along zero or more outgoing links, as shown in Figure 1.1. A link from the ith to the jth unit has a weight w_{ji} which determines the effectiveness with which the ith unit can transmit its output signal to the jth unit. The output of the jth unit, y_j, is computed as some function f of its total weighted summed input x_j:

$$y_j = f(x_j) \tag{1.1}$$

$$x_j = \sum_i w_{ji} y_i + b \tag{1.2}$$

where b is the unit's bias. A convenient way to implement the bias is to treat it as an extra weight connected to an input unit whose activation value is always 1, as shown in figure 1.1. Hereafter, we will omit the explicit bias, b, from the expression for x.

The three most common choices for f are the simple linear function:

$$f(x) = x$$

the binary threshold function:

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

and the sigmoidal or S-shaped logistic function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1.3}$$

Many models assume stochastic binary units, which take on values in $[0, 1]$ (or $[-1, 1]$), and which are randomly activated according to some probabilistic function of their total input (such as the sigmoid function).

The *input units* of a network are those whose pattern of activity I is partially or entirely determined by an environmental or external signal. When their states are entirely determined by the environment, we say that their activities are *clamped* to I. The *output units* are those whose pattern of activity O represents the output of the network. A presentation of an input pattern typically involves clamping the activities of the n input units I to the n elements of the input pattern vector for some period of time, and allowing this activation to propagate along the links, weighted by the connection strengths, to the rest of the network.

A *feed-forward* network is acyclic; the units are arranged in *layers*, so that a unit in layer j receives input from units in layers $i < j$, and sends its output to units in layers $k > j$. We will use the term *single layer network* to refer to a network having a single layer of weights connecting the input units directly to the outputs, while the term *multilayer network* will be used to refer to a network having more than one layer of weights and *hidden units* which are neither input nor output units.

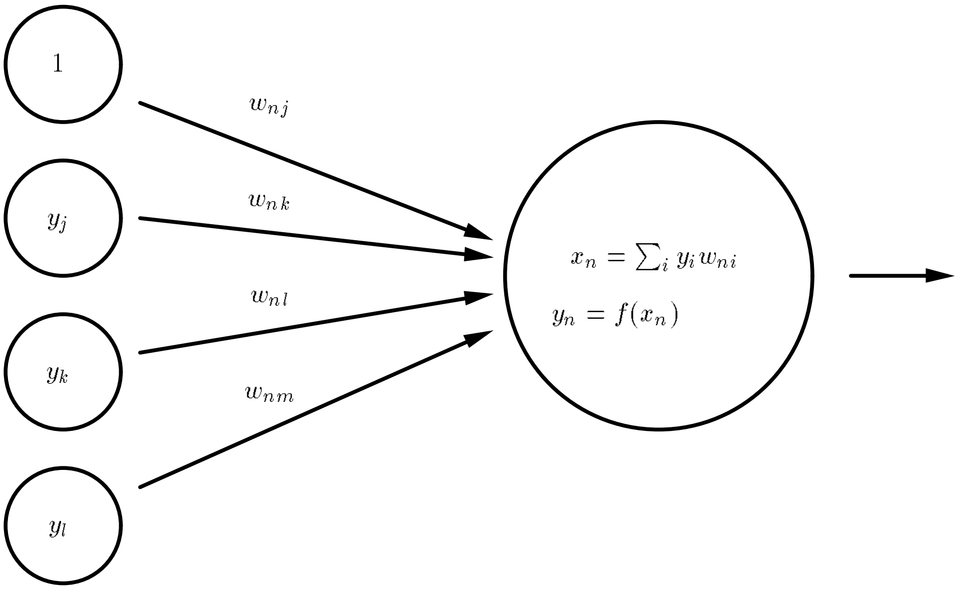

Figure 1.1: *A model neuron that outputs a value which is some function f of its total weighted summed input.*

1.2.2 Measures of information

Several of the learning algorithms derived in this thesis employ measures of mutual information. When we use the term information, we refer to Shannon's information measure. All of the definitions and results in this subsection are due to Shannon (1948), unless otherwise noted. Shannon derived this measure to characterize the amount of information transmitted across a communication channel. Given a random variable with a set of n alternative outcomes with probabilities p_i, Shannon wanted a measure H over the set of probabilities, $H(p_1, p_2, \ldots, p_n)$, with the following properties:

1. H should be continuous in the p_is.

2. For equally likely events, $p_i = \frac{1}{n}$, H should be monotonically increasing in n.

3. For an event i which can be broken down into several alternative events $j, k, l, \cdots$, the original H should be a weighted sum of the individual values of H. More formally, if $0 \leq \lambda \leq 1$, and $\bar{\lambda} = 1 - \lambda$, then $H(p(0), \ldots, p(n-1), \lambda p(n), \bar{\lambda} p(n)) = H(p(0), \ldots, p(n)) + p(n) H(\lambda, \bar{\lambda})$.

Shannon showed that the only H satisfying these three assumptions is of the form:

$$H = -K \sum_{i=1}^{n} p_i \log p_i \tag{1.4}$$

where K is a positive constant representing a choice of units of the measure H. This measure is variously known as the entropy, uncertainty or information content of the set of probabilities. If x is a random variable, $H(x)$ is generally used to refer to the entropy of x (although H is actually a function of $p(x)$, not of x).

For a pair of random variables x and y, if $p(i,j)$ is the joint probability of x taking on the ith possible value and y taking on the jth, the entropy of the joint distribution is:

$$H(x,y) = -\sum_{i,j} p(i,j) \log p(i,j) \tag{1.5}$$

This is often referred to simply as the joint entropy. Similarly, the conditional entropy can be defined, and is often referred to as the equivocation, or the remaining uncertainty in x given y:

$$H(x|y) = -\sum_{i,j} p(i,j) \log p(i|j) \tag{1.6}$$

If x is a transmitted signal and y the signal received, then the rate of transmission of information is defined to be:

$$\begin{aligned} R &= H(x) - H(x|y) \qquad (1.7) \\ &= H(y) - H(y|x) \\ &= H(x) + H(y) - H(x,y) \end{aligned}$$

This defines the amount of information in x less the amount remaining in x when y is known, or in other words, the uncertainty in x which is accounted for by y. Because of the symmetric form of the equation on the third line, this measure is also known as the mutual information between x and y (e.g., (Aczél and Daróczy, 1975)). This quantity is often denoted $I_{x;y}$.

H has a number of desirable properties, including the following:

1. H(x) is zero (minimal) only if all but one of the possible values of x have zero probability. This intuitively corresponds to the situation of maximal certainty in the value of x.

2. H is equal to $\log n$ (maximal) if all the probabilities are equal. This corresponds to the situation of maximal uncertainty.

3. It can be shown that $H(x,y) \leq H(x) + H(y)$, with equality only if x and y are independent.

4. The *Shannon inequality* states that for any two distributions p and q,

   $$-\sum_i p(i) \log q(i) \geq -\sum_i p(i) \log p(i),$$

   with equality only if $p = q$.

For a continuous random variable x with density $p(x)$, the entropy is defined as:

$$H(x) = -\int_{-\infty}^{\infty} p(x) \log p(x) dx \tag{1.8}$$

and the joint and conditional entropies are defined similarly.

For the special case when $p(x)$ is a one-dimensional Gaussian with mean μ and variance σ^2,

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2} \tag{1.9}$$
$$H(x) = \log\left(\sqrt{2\pi e}\sigma\right) \tag{1.10}$$

and for an $n-$dimensional Gaussian with mean μ, and covariance matrix $\mathbf{A}$ having determinant $|\mathbf{A}|$,

$$p(\mathbf{x}) = \left((2\pi)^{n/2}|\mathbf{A}|^{1/2}\right)^{-1} e^{-1/2(\mathbf{x}-\mu)^T\mathbf{A}^{-1}(\mathbf{x}-\mu)} \tag{1.11}$$
$$H(\mathbf{x}) = \log\left((2\pi e)^{n/2}|\mathbf{A}|^{1/2}\right) \tag{1.12}$$

An interesting case to consider, which we will return to in Chapter 4, is the information transmitted through a noisy channel when the source and noise are independent Gaussians. If x is equal to the signal, and y is the signal plus noise, and V is variance, then the information rate, or mutual information between x and y, is:

$$\begin{aligned} R &= 0.5\log\frac{V(signal+noise)}{V(noise)} \\ &= 0.5\log\frac{V(y)}{V(y-x)} \end{aligned}$$

The Shannon inequality gives rise to an error measure introduced by Kullback (1959) as the *asymmetric divergence* between two distributions p and q:

$$G(p,q) = \sum_i p(i)\log\frac{p(i)}{q(i)}$$

This measure, sometimes referred to simply as the G-error, has been used in several neural network learning procedures (e.g. Hinton, Sejnowski and Ackley, 1984; Pearlmutter and Hinton, 1986b) and we will return to it in Chapters 7 and 8.

1.3 Supervised versus unsupervised learning

Connectionist learning paradigms can be roughly divided into two approaches: supervised and unsupervised. In the supervised case, the network is presented with examples of input and desired output patterns, and the goal is to adjust the connection strengths so that the network learns to compute a mapping which best fits the examples, as measured by some optimality criterion (usually the mean squared error between the desired and actual responses of the output units). This paradigm is appropriate when the desired states of the network can be specified exactly, for every input pattern, such as in signal prediction or encoding-decoding problems.

The most successful and widely used supervised learning procedure for multilayer feed-forward networks is called the Back Propagation algorithm (Werbos, 1974; Parker, 1985; Rumelhart, Hinton and Williams, 1986; le Cun, 1987). This algorithm iteratively learns a set of weights by performing gradient descent in a space defined by some error measure. For each input pattern presentation, the partial derivatives of the error with respect to each weight in the network are computed by applying the chain rule in an efficient manner; this involves two passes through the network, one to propagate the states to the output layer, and one to back-propagate accumulated gradient terms from the output units to the input layer. Weights are then incremented by a fixed proportion ε of their partial derivatives, where ε is a global learning rate. Note that the general procedure of back-propagating derivatives of a cost function through intermediary network layers can also be used in unsupervised learning procedures, and we shall return to this idea in later chapters. We shall use the terms "Back Propagation algorithm" and "Back Propagation networks" in reference to the specific supervised learning procedure described above, and the more general term "back-propagation" to refer to the process of accumulating and propagating derivatives backwards through a network.

Back Propagation networks are powerful computational devices. Several researchers have shown that feed-forward networks of logistic units with at least one hidden layer are capable of approximating any continuous function (for a review, see Poggio, 1989). However, the general learning problem for feed-forward nets is NP-hard (Judd, 1987). And even if the learning is allowed to take an indefinitely long time, any learning procedure which is based on a strict descent method has no guarantee of converging to the optimal solution; the best it can do is to find a local minimum in the error function. In practice, however, Back Propagation learning has been successfully applied to some difficult problems. For example, Lang and Hinton (1988) showed that Back Propagation networks outperform Hidden Markov Models (the best currently known statistical methods) on phoneme recognition.

Unfortunately, supervised learning algorithms have so far been limited by their poor scaling behaviour: the learning becomes unacceptably slow as the size of the network or problem increases. This is particularly true of large nonlinear networks with many hidden layers. To understand this, consider that the effect of a weight in the first layer on the output of an m-layer network may (in the worst case) depend on its interactions with on the order of $(fan\text{-}in)^m$ other weights, where *fan-in* is the average number of incoming links of units in the network. Hence, the parameter-tuning problem in a large system with multiple stages of nonlinearities can take a prohibitively long time to solve. Another standard criticism of supervised learning procedures is their lack of biological plausibility, as they are restricted to problems for which an external teacher is available.

One solution to these problems may be to make use of unsupervised learning procedures, which can be applied sequentially, one layer at a time. This would allow deep networks to be trained in time linear in the number of layers. In the unsupervised learning paradigm, rather than providing explicit examples of the function to be learned by the network, we define some task-independent measure of the quality of the representation to be learned, and then optimize the network parameters with respect to that measure. The distinction between supervised and unsupervised learning is not always sharp, and there are some learning paradigms that may better be described as "semi-supervised". For example, in the case of reinforcement learning (Barto and Sutton, 1983), a supervisory "reward" signal is provided which tells the network when it is doing the right thing, possibly on an infrequent basis. Output units are not specifically provided with target values, but nonetheless, they are guided by some external teaching signal. We will not consider such algorithms any further here, but will restrict ourselves to the cases of pure unsupervised learning where the network uses an "internal" measure of its own performance as the basis of learning.[1]

[1] Even this definition could be made more precise, since many unsupervised learning algorithms have been developed which do not refer explicitly to any performance measure; rather, they are defined in terms of an equation for weight updates. One

Unsupervised learning has several computational advantages. Rather than trying to solve difficult problems in one stage by training a large multi-layer network, we can subdivide the problem. First we build adaptive preprocessing modules that can capture some of the regular features in the environment, and form representations having a simpler form than the raw, unprocessed input. If we can define sufficiently general measures of "interestingness," we can then train networks to learn interesting features which should be applicable to many problems (in contrast to the highly problem-specific representations typically learned by supervised nonlinear multilayer networks). Modules can then be assembled hierarchically to extract features of progressively higher order. Finally, once the hierarchy of unsupervised modules has extracted the important underlying *causes* of the sensory inputs, we can add supervised modules on top which can quickly learn to associate responses with representations of the causes of the sensory input, rather than with the input itself.

1.4 Thesis outline

In **Chapter 2**, we review the literature on unsupervised learning. First, we present some neurophysiological evidence of unsupervised learning during perceptual development in young animals. This literature illustrates the advantages of plasticity in developing perceptual processing systems which are specifically tuned to the statistics of the environment. We then review the major approaches which have been taken to modeling unsupervised learning in neural networks. Previous approaches have generally viewed unsupervised learning as an adaptive preprocessing stage for sensory input, the role of this preprocessing being to try to encode *all* of the information contained in the input, while reducing noise, uncertainty, redundancy etc. Hence these approaches are generally limited to modeling only the first few stages of perceptual processing. To move toward the goal of developing higher level representations which will be useful for more abstract processing tasks such as object recognition, we argue that it is necessary to build constraining assumptions about the input into the architecture, and into the learning objective. These assumptions serve two purposes. First, they constrain what information will be extracted by the network, forcing it to extract higher level, more abstract features. Second, they provide a way of decomposing difficult learning problems into several simpler feature-extraction stages. This general approach appears to be a promising way of obtaining more robust and efficient solutions to difficult problems such as object and speech recognition from raw sensory data.

In **Chapter 3**, we show how to derive unsupervised learning procedures by making highly constraining assumptions about the world. We derive an objective function for unsupervised learning based on the assumption of coherence across different parts of the sensory input. Different network modules, looking at different parts of the sensory input, try to extract features which have high mutual information. We first consider the simple case of two binary stochastic units, each receiving input from a different image patch. When applied to a pattern ensemble of sinusoidal intensity patterns, a pair of mutual-information-maximizing units learns to divide up the frequency-phase space into several regions. We then consider the case of discrete n-valued variables. By allocating a group of n units to encode each variable, and interpreting their responses as a probability distribution over the n values, the mutual information between two n-valued parameters can be computed. Two groups of n units receiving input from two neighboring image patches

such example is Hebbian learning, as described in the next chapter. However, a performance measure can often be derived from a weight update equation, by assuming that the learning rule is following the derivative of some objective function.

try to maximize the mutual information between the two n-valued parameters represented by their outputs. When applied to the spatial frequency data, the two groups of units learn to divide up the frequency-phase space more finely, so that each unit responds to a single, continuous region. Next, we show several ways in which the objective can be optimized in multi-layer networks of binary stochastic units: (i) training the layers sequentially, (ii) training the network globally, using back-propagation to train the hidden layers, and (iii) a combined "bootstrapping" training method. We show that the algorithm is able to learn to extract shift in binary shift patterns; this demonstrates that the learning procedure is able to extract a non-trivial, higher-order feature of the input.

Chapter 4 presents a continuous version of the algorithm described in Chapter 3; by making Gaussian assumptions about the features to be learned, we derive a tractable expression for the mutual information measure. We show how this objective can be applied to the problem of learning to extract depth from random dot stereograms of smoothly curved surfaces. This problem, though artificial, illustrates the general principle of learning based on the assumption of spatial coherence in visual images. Neighboring network modules receive input from nearby patches of an image; nearby image patches tend to exhibit similar properties (e.g. texture, colour, depth). So network modules which try to extract common underlying signals from these nearby patches are able to discover features which tend to be roughly constant across space. The first model we consider applies to fronto-parallel planar surfaces. Assuming that the depth in neighboring image patches is roughly equal, pairs of modules try to extract a signal which is common to their inputs. Simulations on different data sets with varying ranges of disparity and random dot densities show that the algorithm is somewhat sensitive to noise in the data; when disparity information is difficult to extract, the learning sometimes becomes trapped in suboptimal solutions. The second model we consider applies to curved surfaces, in which depth varies smoothly across space. Units try to extract a signal which is predictable across *several* nearby image regions. On the stereo problem, this results in network units that learn to interpolate depths of smooth surfaces.

In **Chapter 5** we describe several ways of extending the basic algorithm described in Chapter 4 when we have various mixture models of coherence. In the first model, the parameter of interest is assumed to have a multi-modal distribution. When applied to the continuous stereo problem, again on random dot stereograms of curved surfaces, the algorithm learns to form a population code for depth, in which a set of units are tuned to depth, and each responds optimally to a slightly different range of depth values. Such a code is frequently employed in biological systems, for example, by colour-sensitive photoreceptors in the retina, and by disparity-selective cortical neurons. There are several advantages to this sort of representation, the major one being that if individual neurons have limited dynamic range, then as a group they can encode a wide range of parameter values much more efficiently and accurately. We extend the algorithm described in Chapter 4 to the case of multiple competing units which learn to encode spatially coherent features as a group. We also describe several ways of extending the basic algorithm in order to deal with discontinuities. We use mixture models once again, but this time, in order to separate continuous cases from cases of discontinuities. The simplest method treats the data as being generated by a mixture of two distributions, a predictable, spatially coherent one and an unpredictable, discontinuous one. Network modules attempt to identify which cases fall into the predictable class, and only try to model those cases. A more elaborate method deals more directly with discontinuities, by learning to model them explicitly. When applied to random dot stereograms of curved surfaces with depth discontinuities, this algorithm is able to form a mixture of several different coherence models which apply in different cases, depending on the location of discontinuities. Additionally, the network develops specialized feature detectors which respond to depth discontinuities at different locations. We also present an alternative formulation of the model, using the "competing experts" framework (rather than the mutual-information based algorithm) described by Jacobs et al. (1991).

In **Chapter 6**, we describe an alternative way to discover spatially coherent features in images. We show how a multi-layer network can be trained to discover a higher order feature such as stereo disparity without the need for back-propagation of derivatives, using the more biologically plausible mean field Boltzmann machine learning algorithm. In order to train the Boltzmann machine without supervision, the network is shown two classes of patterns, one spatially coherent and the other incoherent; the network learns to respond differently to the two pattern classes by adopting low energy states for coherent patterns and high energy states for incoherent ones. To achieve these energy levels, it develops disparity-tuned units.

In **Chapter 7** we summarize the major contributions of the thesis, discuss the advantages and drawbacks of each of the algorithms we have described, and suggest directions for future research.

Chapter 2

Background

The mammalian brain undergoes substantial self-organization as the young animal gradually learns to find regularities in the complex patterns of stimuli which it encounters. Eventually a repertoire of environmentally tuned feature detectors is developed, that gives rise to stable, coherent perceptions. Even long after the early critical stages of perceptual development, some of the mammalian brain's representations apparently undergo continual reorganization in response to changing environmental demands. We begin this chapter with a brief review of the neurobiological evidence for self-organization in the brain. This literature motivates our study of unsupervised learning in artificial neural networks, and gives us some insight into how we might employ self-organizing principles in our design of efficient, and perhaps even biologically plausible, recognition systems. We then review the major computational approaches which have been taken to modeling unsupervised learning in neural networks, examine the limitations of this work, and discuss how this literature motivates our research on unsupervised learning.

2.1 Perceptual learning in biological systems

There is substantial evidence that brain development and perceptual functioning are markedly affected by exposure to structured environmental input. Much of the evidence comes from single cell recordings in the primary visual cortex of cats (Area 17) and more recently, cell recordings in the visual and somato-sensory systems of various animals. In several mammals, visual cortical neurons have been found to be tuned to a variety of stimulus features such as orientation (i.e., some cells respond maximally to edges or bars of particular orientations), spatial-frequency, velocity and direction of movement, and binocular disparity. The fine-tuning of the visual system to many, if not all, of these features occurs primarily during certain "sensitive periods of development" early in life, and can be disrupted if an animal is reared in an impoverished environment.

When kittens are raised in environments containing only vertical or horizontal contours, cortical cell populations exhibit orientation selectivities which are skewed in favor of those particular contours to which the animal was exposed (Blakemore and van Sluyters, 1975; Blakemore and Cooper, 1970; Hirsch and Spinelli, 1970). It has been pointed out that since orientation tuning is present in visually inexperienced kittens (Hubel and Wiesel, 1963), the effect of distorted environmental input may not be to actually alter the orientation preferences of single cells, but may be to merely cause atrophy of other cells from disuse (Miller, 1990). However, a study by Frégnac et al. (1988) showed that shifts in both the orientation and ocular preferences of individual cells can be induced by pairing visually presented stimuli with artificially induced cell firing; shifts of up to ninety degrees in orientation preferences were reported. A study by Spinelli and Jensen (1979) further supports the hypothesis that orientation selectivity of cortical cells can be learned, even in normally reared kittens. Kittens were placed in a controlled visual environment for 8 minutes per day, during which time a striped pattern presented to one eye was paired with electrical stimulation of the forelimb (strong enough to elicit paw withdrawal); the kitten could turn off the shock by raising its right paw, at which time an orthogonally oriented striped pattern was presented to the other eye. Kittens quickly learned the task. The rest of the time, they were in a normal environment, with their mother and siblings. After about 10 weeks of training, cell recordings from the primary visual cortex of these kittens revealed that many cells responded to vertical stripes in one eye *and* horizontal stripes in the other (not necessarily simultaneously presented). This effect is difficult to explain in terms of an "atrophy from disuse" argument; it seems that these cells became tuned specifically to the unique regularities of the kittens' environment. It is noteworthy that although the behavioural salience of the task increased the likelihood of the unusual cell types being found, similar patterns were found, albeit to a lesser degree, in yoked control kittens, who were presented with the same visual stimuli without any contingency upon shock or limb withdrawal. This suggests that although reinforcement magnifies the plasticity of the visual system, some degree of perceptual learning takes place entirely unsupervised.

Spatial-frequency tuning of feline cortical cells is also influenced by early visual experience: while no visual input is necessary for normal development of spatial-frequency selectivity during the first three weeks, the subsequent fine-tuning of frequency sensitivity which normally occurs in weeks 4-8 is blocked if the kitten is deprived of patterned input (Derrington, 1984). Another widely studied developmental phenomenon in the cat's visual cortex is the appearance of ocular dominance columns (ODCs); after normal binocular visual experience, cells in primary visual cortex which receive input from both eyes eventually segregate into alternating eye-specific patches. This effect can be strongly influenced by environmental factors: when a cat is subjected to monocular deprivation, particularly during the second postnatal month, cortical cells become almost exclusively and irrecoverably responsive to input from the exposed eye (Wiesel and Hubel, 1965; Olson and Freeman, 1980). In contrast, if signals from *both* eyes are prevented from activating cortical cells, either by blocking retinal activity (Stryker and Harris, 1986) or cortical activity (Reiter, Waitzman and Stryker, 1986), no ocular segregation occurs at all in the cortex. Thus, it appears that through normal visual experience, cortical cells become tuned to the spatially correlated inputs from one eye or the other; further, through some sort of competitive/co-operative interaction, they tend to form islands of similarly tuned cells (ODC's).

There is similar evidence of activity-dependent development in primate perceptual systems. For example, monkeys with optically induced strabismus (a misalignment of the two eyes, either convergent or divergent) early in life, followed by three years of normal binocular experience, have a marked reduction in the number of cortical binocular cells (in both areas V1 and V2) and are behaviourally stereoblind (Crawford et al., 1984). In humans, stereopsis emerges abruptly at about the end of the fourth month of life (Held, Birch and Gwiazda, 1980; Birch, Gwiazda and Held, 1982), and stereoacuity continues to improve during the first year of life (Birch, Gwiazda and Held, 1982). Children with esotropia (a convergent strabismus, commonly known as being "cross-eyed") in infancy, which is surgically corrected between 10-13 years of age, perform identically to strabismic monkeys on stereopsis tasks – they are clinically stereoblind (Crawford et al., 1983).

Stereoblindness may be preventable, however, by early intervention. Deficits associated with stereoblindness, including strabismus (Birch et al., 1990) and congenital unilateral infant cataract (Maurer and Lewis, 1992) can be corrected by a combination of surgery and occlusion therapy on the good eye (to eliminate monocular fixation preferences); if applied during the first year of life, these treatments are associated with a potential for the development of at least gross stereopsis. This suggests the possibility of a similar critical period in humans for the development of binocular vision.

Cortical reorganization occurs not only during sensitive periods early in development, but continues through the lifetime of an animal, as a result of changing environmental demands. For example, Merzenich and his associates (Merzenich, 1987; Merzenich et al., 1988) have studied plasticity in the somato-sensory cortex of the adult owl monkey; neurons in this region of the brain respond to tactile input from local areas of the skin surface, forming a roughly topographic map of the body. Merzenich's group has extensively studied cells whose receptive fields correspond to regions of skin surface on the hand (in Area 3B); these receptive fields change drastically when the spatio-temporal correlations of tactile stimulation are experimentally varied.

It is possible that cortical cells employ some very general organizing principles which are independent of the sensory modality, in developing their characteristic responses to patterned input. Startling evidence for this possibility comes from an unusual study in ferrets done by Sur, Garraghty and Roe (1988). Nerve fibers from the primary visual pathway were artificially redirected into the auditory cortex, and these visual inputs formed synaptic contacts with cells in the auditory cortex. After the ferrets were reared to adulthood, the majority of "auditory" cortical cells tested had become visually driven, some having center-surround receptive fields[1] similar to cells in the visual cortex.

The ability to develop a set of environmentally tuned feature detectors is of clear adaptive advantage. The organism need not have a completely genetically predetermined perceptual system; rather, the primitive organization laid out by the genetic blueprint can be fine-tuned after birth, as the statistics of the environment, the organism's behavioural requirements, and even the physical properties of the organism's own sensors (e.g., the distance between the eyes, ears etc.) change over time. Once a set of environmentally tuned feature detectors has been learned, important features of the world such as local variations in position and depth can be quickly extracted by individual cells or cell populations in parallel. The response of these cells can then serve as a preprocessed input to higher layers, which can in turn self-organize to form still more complex representations.

2.2 Computational models of unsupervised learning

Motivated by the evidence of activity-dependent self-organization in the brain, described above, and by the poor scaling behaviour of supervised learning procedures (discussed in Chapter 1), our goal is to discover ways in which artificial systems can perform unsupervised learning. By discovering good solutions to this problem in simulated networks of neuron-like processing elements, we hope to cast light on how the brain may solve the same problem. Further, we would like to apply the general principles employed in biological systems to help build more efficient artificially intelligent systems: rather than trying to solve difficult

[1]A cell having a "center-surround" receptive field tends to respond maximally to illumination in the centre of its receptive field and to be inhibited by illumination in the periphery ("on-centre off-surround") or vice versa ("off-center on-surround").

problems in one stage by training a large multi-layer network, we can subdivide the problem. First we build adaptive preprocessing modules that can capture some of the interesting features in the environment, and form representations of a simpler form than the raw, unprocessed input. Modules can then be assembled hierarchically to extract features of progressively higher order. Finally, once the hierarchy of unsupervised modules has extracted the important underlying causes of the perceptual inputs, a supervised module can quickly learn to associate responses with the causes of the perceptual input rather than the input itself.

2.2.1 Hebbian learning

The earliest proposal for an explicit rule of synaptic modification was made by Donald Hebb (1949). Hebb's now famous postulate about the mechanism of learning is as follows:

> *When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased. (Hebb, 1949)*

Now widely referred to simply as Hebbian learning, this rule has been used as the basic building block of associative memory in a wide variety of unsupervised learning models, as we shall see. The most direct computational expression for Hebb's rule causes the strength of a weight w_{ji} from unit i to unit j to increase in direct proportion to the product of presynaptic and postsynaptic activities:

$$w_{ji}(t) = w_{ji}(t-1) + \varepsilon y_i(t) y_j(t)$$

One problem with this rule is that, assuming activities are always positive, the connection strengths can increase without bound. Many other forms of Hebbian learning rules have been proposed which attempt to model various aspects of associative learning (e.g. Sutton and Barto, 1981; Bienenstock, Cooper and Munro, 1982; Kosko, 1986; Linsker, 1986a,b,c; Tesauro, 1986; Klopf, 1987; Sejnowski and Tesauro, 1989); we describe two of these models in some detail.

Bienenstock, Cooper and Munro (1982) proposed a variation of the simple Hebbian learning rule which results in a form of temporal selectivity. The basic idea is that in order for a single unit to compute some useful function it should respond selectively to some input patterns and not others, with respect to some particular environment. They proposed the following measure of selectivity, which depends on the ratio of the unit's mean response over all inputs to its maximal response:

$$\mathrm{Sel}(y_j) = 1 - \frac{\overline{y_j}}{\max(y_j)}$$

The ideal unit, by this measure, gives a maximal response to one particular pattern, and very low responses to the other patterns. To achieve this, the authors proposed a family of Hebb-like learning rules which cause a weight w_{ji} to change in proportion to the product of the presynaptic activity y_i for that link and some function Φ of the postsynaptic activity y_j. Φ is chosen so that the sign of the weight change reverses when y_j is sufficiently large relative to the mean response $\overline{y_j}$. The general form of their learning rule is:

$$\dot{w}_{ji} = \Phi(y_j, \overline{y_j}) y_i - \epsilon w_{ji}$$

where $\dot{w}$ denotes the rate of change of w over time. The rightmost term causes weights to exponentially decay toward zero in the absence of inputs. The function Φ must be chosen so that the unit's state converges to some stable equilibrium point of high selectivity. The authors show that Φ must therefore have the following properties:

$$\mathrm{sign}\left(\Phi(y_j, \overline{y_j})\right) = \mathrm{sign}\left(y_j - \left(\frac{\overline{y_j}}{c_o}\right)^p \overline{y_j}\right) \quad \text{if } y_j > 0$$

$$\Phi(0, \overline{y_j}) = 0 \quad \text{for all } \overline{y_j}$$

where p and c_o are positive constants, and units' activities are always positive. This says that when the unit's response is uniformly low ($\overline{y_j} \ll c_o$), the weight changes for almost all patterns should be positive. If the mean response is very large ($\overline{y_j} \gg c_o$), on the other hand, all the weight changes should be negative. At some intermediary point ($\overline{y_j} \approx c_o$), a stable state is reached such that all the weight changes are zero; at this point, all the responses are either pinned near their minimum values or near $\left(\frac{\overline{y_j}}{c_o}\right)^p$. Some interesting results were reported for the development of tight orientation tuning curves for units, using simple oriented line patterns as inputs.

One problem with this approach is that the criterion of maximal selectivity cannot be directly optimized since it is not a continuous function. The authors experiment with a number of learning rules with different choices of the function Φ (e.g., continuous versus discontinuous, with different values of p and c_o) which satisfy the general form given above. Unfortunately, not all choices of Φ necessarily result in learning rules which converge to high selectivity solutions. For example, choosing p close to zero and Φ continuous would result in a relatively flat selectivity curve. Intrator (1990) has proposed an objective function for maximizing selectivity which is related to the *skewness* of the distribution. This causes a unit to discover projections of the data having bimodal distributions. Further, he shows how this objective, when applied to a group of units which inhibit each other, leads to the discovery of multiple features in the data.

Linsker (1986a,b,c) developed a slightly more complicated Hebbian rule for networks of units with spatially localized receptive fields. In Linsker's model, each layer of linear units is arranged in a two-dimensional grid, and a unit in layer L is connected to N_M of the units in the preceding layer M, with the probability of a connection falling off as a Gaussian function of the distance from the L unit's centre. The following Hebbian learning rule is used:

$$\Delta w_{kj}{}^{\alpha} = c_1 + c_2(y_k^{M\alpha} - c_3^M)(y_j^{L\alpha} - c_3^L)$$

where the c_i's are constants, the L and M superscripts denote layers, and the α superscript denotes a particular training pattern. The above rule can be averaged over an ensemble of training patterns, assuming the time course of learning is much longer than the time interval of a pattern ensemble. A weight on the ith input line to a unit changes in proportion to its covariance Q_{ij} with every other input j to the same unit:

$$\dot{w}_{kj} = k_1 + \frac{1}{N_M}\sum_i (Q_{ij}{}^L + k_2) w_{ki}$$

where k_1 and k_2 are more constants. Linsker experiments with different choices of the four parameters k_1, k_2, N_M and $\frac{r_M^2}{r_L^2}$ (the ratio of the areas of receptive fields from layer L to M). By fixing these parameters, and limiting the values of weights to lie within the interval $[-1, 1]$, the evolution of the weights to a unit in a given layer can be simulated, provided the covariance matrix for unit activities in the previous layer are known. Linsker uses random inputs, so the covariance between each pair of input units is zero. For the input layer, the parameters k_1 and k_2 can be chosen so that these weights become "all-excitatory" (i.e., they are pinned at their positive extrema). Once a layer of weights is fixed, the correlation function for units in the next layer can be computed. Whereas units in the input layer are uncorrelated, units in the next layer have overlapping receptive fields which may share some inputs; hence, these units will be partially correlated, with correlation falling off exponentially as the distance between units' centers. Now the maturation process can be simulated for each successive layer L, by choosing values for the four parameters, computing Q^L, and then solving the ensemble-averaged equation for the weights to that layer.

Linsker (1986c) finds that with appropriate choices of the learning constants, units in the intermediate layers evolve a progressively more "Mexican hat" shaped receptive field, much like the "on-centre, off-surround" and "off-centre, on-surround" receptive fields found in the early levels of mammalian visual systems. When the process is repeated for several more layers, again with appropriate selections of the four parameters at each layer, cells can develop into orientation specific banded receptive fields, much like the simple cells in primary visual cortex (Linsker, 1986a). To explain these results, Linsker derives an energy function, in which his learning rule performs gradient descent. He reports that the minima of this energy function found by the method of simulated annealing correspond closely to those found in his network simulations. Yuille, Kammen and Cohen (1989) have shed further light on Linsker's results by analyzing a closely related energy function. They show analytically that for inputs obeying certain forms of spatial correlation functions (e.g., the Laplacian of a Gaussian with a small asymmetry), the minimum energy solutions correspond to oriented receptive fields. The addition of lateral inhibition leads to oriented quadrature pairs.

Linsker's model has limited computational power, since he uses an entirely linear system; this could be collapsed into into a single linear operator, expressible by a single layer of weights.[2] Another drawback to Linsker's model is that many parameters must be hand-tuned for each layer to produce the desired receptive fields at the top layer. However, in spite of these drawbacks, Linsker's work is important for several reasons. First, it shows how the simple learning rule proposed by Hebb can be applied in multiple stages of learning, giving rise to interesting classes of feature detectors. Second, it demonstrates a mechanism by which feature detectors found in animal visual systems could develop, given purely random input and appropriate architectural constraints (i.e., spatially localized receptive fields with a Gaussian connectivity distribution). This raises the interesting possibility that a similar process of perceptual learning could occur before birth; during this time, although patterned input is unavailable, spontaneous firing of photoreceptors may occur. Barrow (1987) has shown that the same principles could operate on patterned input; he obtains similar results to Linsker's using natural images (after low-pass filtering and Gaussian windowing) as training patterns, in a Hebbian network of units with lateral interactions. Miller, Keller and Stryker (1989) have performed similar simulations with units having binocular inputs and local lateral excitatory connections; using a Hebbian learning rule with decay in this case leads to the formation of ocular dominance columns.

[2] However, Geoff Hinton (personal communication) has pointed out that the weight matrix for each layer in Linsker's model is very sparse and highly structured, due to the local connectivity. This property would be lost if the levels were collapsed, so learning the same operator in a single layer might be more difficult.

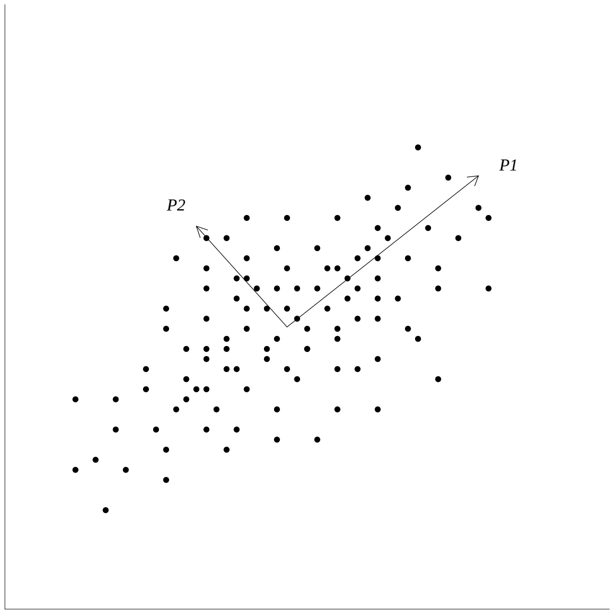

Figure 2.1: *A two-dimensional data distribution. The arrows,* P1 *and* P2*, indicate the directions of principal variation.*

2.2.2 Algorithms that compute principal components

Many of the early models of unsupervised learning were based on ad-hoc learning rules which were determined empirically to produce interesting behaviour. A more principled approach is to first postulate an objective function for the learning, and from that, derive the weight update equations; this provides a theoretical motivation for the model, and makes it easier to analyze the learning procedure and predict its behaviour. One possible objective for unsupervised learning is to discover the principal components of the input distribution, or the N most significant components (i.e., the eigenvectors of the input correlation matrix corresponding to the N largest eigenvalues). Figure 2.1 illustrates the directions of principal variation for a two-dimensional distribution. In this section, we review several neural network learning algorithms which are related to Principal Components Analysis (PCA). These algorithms learn a set of linear orthogonal projections in the directions of principal variation in the input distribution, or a rotated subspace of these directions.

Oja (1982) analyzed a version of the simple Hebbian learning rule for a single unit, which rescales each updated weight to maintain the Euclidean norm of a unit's weight vector equal to one:

$$w_{ji}(t) = \frac{w_{ji}(t-1) + \varepsilon y_i(t) y_j(t)}{\left\{\sum_{k=1}^{n} \left[w_{jk}(t-1) + \varepsilon y_k(t) y_j(t)\right]^2\right\}^{\frac{1}{2}}}$$

For sufficiently small ε, this can be approximated by the following rule:

$$w_{ji}(t) = w_{ji}(t-1) + \varepsilon y_j(t) \left(y_i(t) - y_j(t) w_{ji}(t-1)\right)$$

This rule causes each weight to grow in proportion to the product of the presynaptic and postsynaptic responses, as in the usual Hebbian rule, but adds an internal feedback term which limits the rate of growth in proportion to the size of the output on each case. Oja proved that this learning rule results in a unit that computes the first principal component of the input, i.e., its weight vector converges to (ignoring sign) the principal eigenvector $\vec{w}^*$ of the input correlation matrix, provided that the initial weight vector $\vec{w}(0)$ is not orthogonal to $\vec{w}^*$. Viewed another way, this rule results in a unit which maximizes the variance of its output, subject to the constraint that the norm of its weight vector is one (Sanger, 1989c).

Generalizing this idea to the case of N units, Oja (1989) considered a more complicated learning rule in which the internal feedback term for a particular unit depends not only on the output of that unit, but on a weighted combination of the activities of all the other units in the output layer:

$$w_{ji}(t) = w_{ji}(t-1) + \varepsilon y_j(t) \left(y_i(t) - \sum_{k=1}^{N} w_{ki}(t-1) y_k(t) \right)$$

This rule says that a weight on a given connection should grow as the product of the presynaptic and postsynaptic responses, but that this effect should be reduced in proportion to the output values of other units which are connected to that same input unit, weighted by their connection strengths for that input. Oja called this type of learning network the Subspace Network, because it converges to a set of k weight vectors which span the same subspace as the coefficient vectors of the first k principal components of the input distribution (Oja, 1989).

Sanger's Generalized Hebbian Algorithm (GHA) (Sanger, 1989a,b,c), allows a group of linear units to learn an N column weight matrix whose columns converge to precisely the first N principal eigenvectors of the input correlation matrix, in descending eigenvalue order. Combining Oja's normalized Hebb rule (which computes the first principal component) with Gram-Schmidt orthogonalization, GHA employs the following learning rule:

$$w_{ji}(t) = w_{ji}(t-1) + \varepsilon y_j(t) \left(y_i(t) - \sum_{k=1}^{j-1} w_{ki}(t-1) y_k(t) \right)$$

Note that this rule looks remarkably like Oja's subspace learning rule, except that in the GHA, the ith unit's weight updates depend on feedback only from the $j-1$ preceding units in the sequence (hence the inherent sequentiality of Sanger's algorithm), rather than *all* of the other units. So we can think of Oja's algorithm as creating a sort of "push and pull" process among all of the units to attain orthogonal weights while each tries to account for as much variance as possible; in contrast, Sanger's algorithm designates one particular unit to compute the first principal component by accounting for as much of the variance as possible, another to compute the second component by remaining orthogonal to the first unit and computing the projection that accounts for as much as it can of the remaining variance, another to compute the third (by remaining orthogonal to the first two), etc.

One problem with the GHA is that its method of finding successive eigenvectors is numerically poor, so it is best suited to finding only the first few eigenvectors of high-dimensional data (Sanger, 1989c). On natural images, Sanger reports that these components are similar to the centre-surround and oriented feature detectors discovered by Linsker's (1986a,b,c) and Barrow's (1987) networks. This may limit the algorithm's ability to model the full range of feature detectors (e.g., at varying spatial frequencies) found in primate visual systems; when applied to natural images, it "does not discover filters with different frequency response until very late in eigenvalue order, and it requires many iterations to produce poor-quality results" (Sanger, 1989c).

2.2.3 Pattern encoding procedures

Another approach to unsupervised learning is the "encoder paradigm". The main goal is to store a collection of N patterns so that they can be recalled as accurately as possible. Other potential benefits of the process are data compression, feature discovery, pattern completion (i.e., retrieval of one of the stored patterns when only part of it is supplied), and generalization to patterns which were not in the training set. In order to achieve good generalization, the hidden units of the network must discover interesting features of the input distribution. For this reason, the representation formed by the hidden layer of encoder networks is of primary interest in the context of our discussion on unsupervised learning.

The Back Propagation algorithm can be used within the encoder paradigm to train a network to learn the identity mapping, by making the desired states of the N output units identical to the states of the N input units on each case (Hinton, 1987). We refer to this type of network as an *auto-encoder* (see Figure 2.2). Typically some number of hidden units $M < N$ is used. By passing the input through this "bottleneck", some degree of data compression is achieved. The network can potentially learn to encode interesting features of the input patterns, by being forced to find a less redundant code with a small number of hidden units. Some generalization can therefore be expected, since patterns which were not in the training set should produce codes which are somewhere in between the codes of the closest patterns in the training set. Thus, as in supervised Back Propagation, the network should learn some sort of smooth interpolation over the patterns in the training set. (Note that an autoencoder network would have limited pattern completion capabilities, since a partial input pattern would be expected to cause roughly the *average* of several stored patterns to be produced at the output layer, rather than the *nearest* stored pattern.)

Zipser (1986) has shown that nonlinear auto-encoder networks can discover interesting representations of image features, using very simple two-dimensional images of points blurred through a Gaussian, and then discretely sampled, as illustrated in Figure 2.2. When each image consists of a single Gaussian spot in varying positions, and two hidden units are used to encode the ensemble of images, the hidden units form an orthogonal code for 2D position (which in this simple case is sufficient to accurately reconstruct the entire image). As the number of hidden units is increased, there is a gradual transition from a "variable code" in which spot location is a direct function of hidden unit activity, to a "value code" in which collections hidden units' responses together encode the space of 2D coordinates. In the latter case, the hidden units' responses form complicated, roughly oscillating receptive field patterns which overlap to varying degrees. Zipser also applied the autoencoder to the "stereo" task of reproducing input patterns consisting of pairs of one-dimensional images of Gaussian spots at varying disparities. When four or more hidden units were used, there was some degree of disparity between the two dots in the reconstructed image. Interestingly, with four hidden units, rather than encoding the two input images independently by position with two hidden units devoted to each image, a binocular or depth code was always learned by each hidden unit.

Cottrell, Munro and Zipser (1987) studied the effectiveness of nonlinear auto-encoder networks in performing image compression. They trained networks to learn identity mappings on inputs consisting of small patches of a large image, with varying degrees of compression (achieved by decreasing the number of hidden units and by quantizing the hidden units' outputs by different amounts). They observed that hidden units tended to learn roughly equal variance encodings (clearly not a principal components decomposition), and suggested that this is because Back Propagation tends to distribute the error fairly evenly to the hidden units. This might also account for Zipser's finding that hidden units in his stereo network always learned

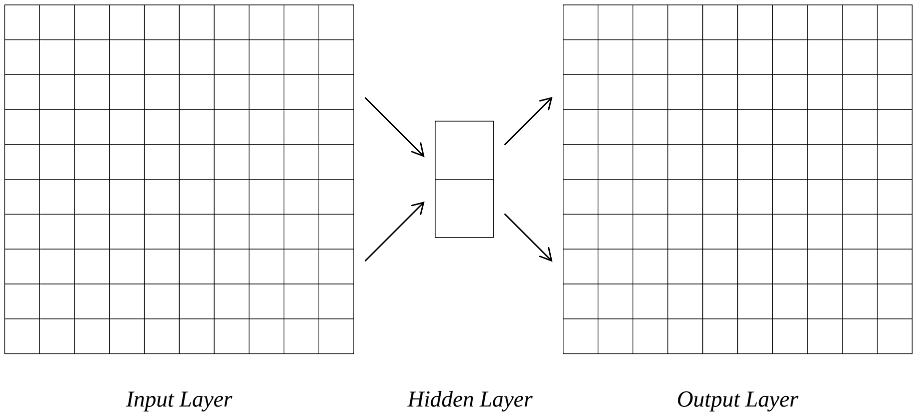

Figure 2.2: *The auto-encoder architecture used by Zipser (1986) to encode the 2D position of a Gaussian spot.*

features common to both images rather than dividing themselves among the two images and learning separate encodings. The nonlinear autoencoder was compared to a purely linear autoencoder; interestingly, the latter gave much better generalization performance (with respect to mean squared reconstruction error on a test set) when applied to new images. The authors' explanation for this effect is that no quantization was applied to the outputs of the hidden units in the linear case, so they had a much larger dynamic range with which to represent images. A second possibility is that the network could have learned a nonlinear function which exactly fits (i.e., overfits) the training set, but generalizes much more poorly than would a smoother function.

An interesting open question remains: what advantages, if any, do nonlinear neural networks have over linear encoders (including PCA) with respect to data compression, quality of image reconstruction, and the nature of the learned representation? In a purely linear network, the auto-encoder paradigm is equivalent to PCA with respect to computational power, since a linear Back Propagation autoencoder converges to a set of weight vectors which is a linear combination of the eigenvectors of the input correlation matrix (Baldi and Hornik, 1989). In the nonlinear case however, as mentioned earlier, Zipser showed that autoencoders could learn rather interesting representations, particularly as the number of hidden units was increased.

As mentioned above, another desirable property of an encoding network is pattern completion – the ability to fill in missing parts of a previously stored pattern, and clean up noise. Good pattern completion generally requires iterative retrieval, and can be achieved to varying degrees by networks with feedback, such as Hopfield networks (Hopfield, 1982), Boltzmann machines (Hinton and Sejnowski, 1986), and recurrent Back Propagation networks (Almeida, 1987; Pineda, 1987; Simard, Ottaway and Ballard, 1989), by creating basins of attraction around the stored patterns in "state-space".

A Hopfield network (Hopfield, 1982) consists of binary threshold units arbitrarily interconnected. Units asynchronously update their states so as to minimize the global energy of the network:

$$E = -\sum_{i<j} w_{ji} x_i x_j$$

A unit is repeatedly selected at random, and its output y_i is set to the on-state if its total input is positive, $\sum_i w_{ji} x_i > 0$, and the off-state otherwise. Hopfield showed that if the network is symmetrically connected, such that $w_{ji} = w_{ij}, \forall i, j$, this update procedure converges to a local energy minimum. Hopfield suggested a Hebbian "one-shot" (non-iterative) learning rule to store an ensemble of M patterns:

$$w_{ji} = \frac{1}{M} \sum_{\alpha=1}^{M} y_i^\alpha y_j^\alpha$$

While the Hopfield net has interesting theoretical properties, it is of limited practical importance because of its low memory capacity, its tendency to become trapped in local minima, its tendency to form spurious attractors not associated with any patterns, and the fact that its learning procedure cannot be applied to networks with hidden units. The Boltzmann machine (Hinton and Sejnowski, 1986) overcomes the problem of local minima by replacing the binary threshold units of Hopfield nets with binary stochastic units, and using a simulated annealing procedure; this allows it, theoretically (if the network is permitted to cool infinitely slowly to a temperature of zero), to settle to the global energy minimum, and to represent probability distributions of network states at equilibrium. Each unit flips states randomly; the probability of a unit adopting the on-state depends on both the contribution of its state to the global energy (using the same energy function as for the Hopfield net), and the temperature of the system, T:

$$p_i = \frac{1}{1 + e^{-\triangle E_j / T}}$$

where $\triangle E_j = \sum_i w_{ji} x_i$ is the difference in the global energy when the jth unit is turned off and turned on. Simulated annealing is performed by starting the system at a high temperature and gradually cooling it, allowing the network to settle to equilibrium at each temperature. At high values of T, units are very likely to flip states randomly, whereas at low temperatures, state changes become more closely tied to the minimization of E. This update procedure causes the state of the network to converge to the Boltzmann distribution, in which the relative probability of any two states α and β is directly related to their energy difference:

$$\frac{P_\alpha}{P_\beta} = e^{-(E_\alpha - E_\beta)/T}$$

The problems associated with learning in Hopfield networks (spurious attractors, low capacity) are overcome in the Boltzmann machine by using a two-phase learning procedure. In the positive phase, the visible units are clamped, the network settles to equilibrium, and the weights are adjusted so as to increase the relative probability of the network being in its current state. In the negative ("unlearning") phase, none or only a subset of the visible units are clamped, the network settles, and the weights are adjusted so as to *decrease* the relative probability of being in this state. The network actually minimizes the G-error, or divergence (as defined in Chapter 1) between the probabilities of states in the positive and negative phases:

$$G = \sum_\alpha P^+(V_\alpha) \log \frac{P^+(V_\alpha)}{P^-(V_\alpha)}$$

where $P^-(V_\alpha)$ is the probability of the visible units being in state α during the negative phase, $P^+(V_\alpha)$ is the probability during the positive phase, and α indexes over all possible states of the set of visible units. The network thereby learns to adopt the same distribution of states in the negative phase which were clamped in the positive phase. The fact that the network first settles to equilibrium on each learning iteration ($\frac{\partial E}{\partial y_j} = 0, \forall j$) leads to a simple Hebbian weight update rule, which depends only on the average of the products of the presynaptic and postsynaptic states in the two phases:

$$\frac{\partial G}{\partial w_{ji}} = -\frac{1}{T}\left[\langle s_i s_j \rangle^+ - \langle s_i s_j \rangle^-\right]$$

where $\langle s_i s_j \rangle^+$ represents the expected value of the products of the states of the ith and jth units in the positive phase, and $\langle s_i s_j \rangle^-$ represents the same quantity in the negative phase. With the addition of hidden units, the Boltzmann machine can theoretically learn to represent arbitrary probability distributions.

Typically the Boltzmann machine is used for supervised learning, by designating some of the units as inputs and some as outputs, although this need not be the case. The Boltzmann machine can be trained as an autoencoder (but without the need for explicit back-propagation of gradients) by allocating equal numbers of input and output units, and making the target outputs identical to the inputs (see figure 2.3 a). Ackley, Hinton and Sejnowski (1985) showed that it could learn to efficiently encode simple binary patterns using the minimal number of hidden units. Unfortunately, the stochastic sampling required to anneal and to accurately estimate the product statistics for the weight updates makes learning in Boltzmann machines prohibitively slow. However, using the mean field approximation, it can be run deterministically (Peterson and Anderson, 1987; Hinton, 1989) and learning is then much faster.

Peterson and Hartman (1989) suggested another way to train the Boltzmann machine in an unsupervised mode, to act as a "Content-addressable memory" (CAM). They used a deterministic Boltzmann machine (DBM) having a single set of visible units (eliminating the distinction between inputs and outputs, as in figure 2.3 b). The goal was simply to train the network to store patterns presented to the visible units, and perform pattern completion. During the positive phase, all of the visible units were clamped to the desired pattern. In the negative phase, a random subset of the units were left unclamped. The network was able to learn a set of random patterns, and perform pattern completion when partially specified or noisy patterns were presented. Further, when no connections were allowed within the hidden layer, the storage capacity was found to be substantially greater than that of Hopfield nets, and to scale linearly with the number of hidden units (Hartman, 1990).

Freund and Haussler (1992) described yet another way to train Boltzmann machines unsupervised. The goal was to learn the "hidden causes" of a collection of patterns, with each hidden unit representing one hidden cause. They used a restricted architecture, allowing only between-layer connections (see figure 2.3 c), so that when all of the visible units are clamped, no settling is required. For this restricted Boltzmann machine, they derive an algorithm for efficiently computing the absolute probabilities of each input state, summed over every possible state of the hidden units. This leads to a single-phase learning procedure which maximizes the absolute probability of training patterns. The network thereby learns to adopt hidden unit states which are good generators of the input pattern set, that is, which detect correlations in the pattern ensemble. This type of model is potentially useful for learning features of the input for later classification, and also for doing pattern completion. However, as the model has only a single hidden layer, it could take exponentially many hidden units to model an arbitrary probability distribution (Radford Neal, personal communication).

Neal (1992) presents a more general way of modeling the probability distribution of a pattern set in a multilayer stochastic network, which falls within the class of Judea Pearl's belief networks (Pearl, 1988). Neal's connectionist belief networks are similar to stochastic Boltzmann machines in that the probability of

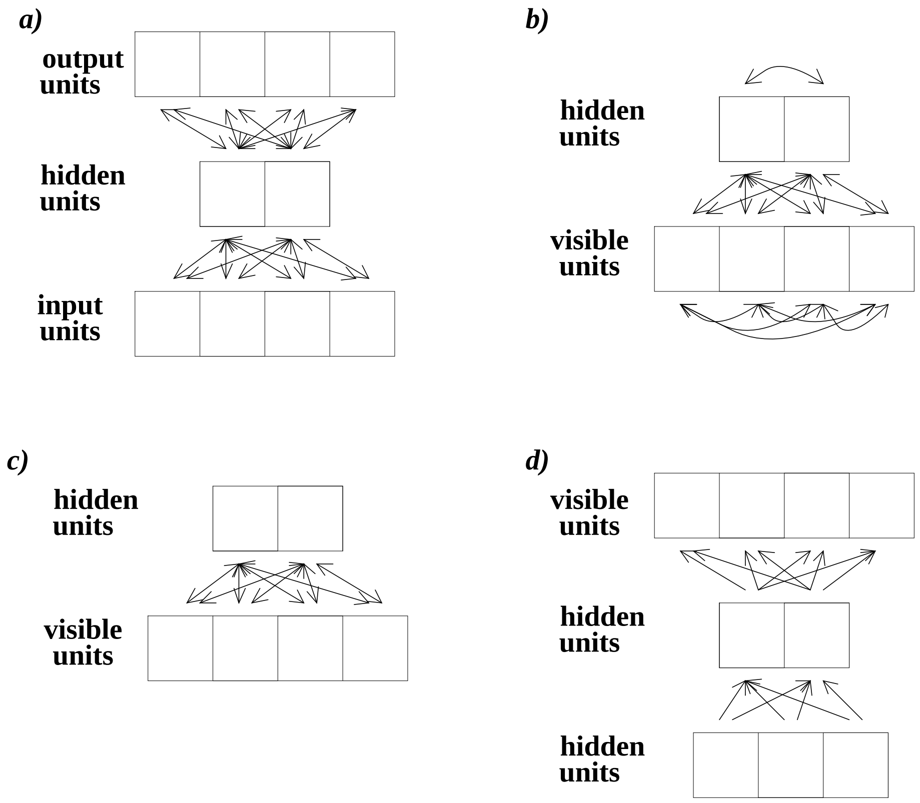

Figure 2.3: *Four general architectures used for pattern encoders (the actual number of units used in the reported simulations may vary): a) A 4-2-4 auto-encoder Boltzmann machine used by Ackley, Hinton and Sejnowski (1985); b) A DBM-CAM, proposed by Peterson and Hartman (1989); c) Freund and Haussler's (1992) restricted Boltzmann machine, same as c) but no within-layer connections; d) Neal's (1992) connectionist belief network.*

a unit adopting the on-state is related to a sigmoidal function of its total input; however, the connections are strictly feedforward (see figure 2.3 d). Output states are clamped to patterns selected from the environment, while the hidden unit state space is randomly explored. The weights are adjusted so as to increase the probability of the hidden units generating the clamped output patterns. The network thereby learns to represent features in the hidden layer which explain correlations in the pattern set, as does the Freund and Haussler model. Both methods can learn sets of hidden variables which *jointly* explain the data. Neal's method is more general, however, as it is not restricted to single layer networks. It should therefore be able to model complex distributions with fewer parameters, although with a high price in learning time.

2.2.4 Algorithms that perform clustering

Many algorithms have been developed for unsupervised learning in artificial neural networks which are variants of statistical clustering algorithms, and are generally referred to as competitive learning procedures (von der Malsburg, 1973; Fukushima, 1975; Kohonen, 1982; Carpenter and Grossberg, 1983; Rumelhart and Zipser, 1986). The basic idea underlying competitive learning is to have some sort of competition between units' responses, so that only one unit in each competitive cluster tends to become active for each input pattern or pattern class. If a limit is imposed on the number of patterns each unit can respond to, units will tend to partition themselves among the input patterns fairly evenly.

One way to induce a winner-take-all competition is via lateral inhibitory connections (i.e., negative links) between units within a layer; such a mechanism was originally proposed by Rosenblatt (1958) in one of his perceptron models. In von der Malsburg's early competitive learning model (1973), a spatially localized version of this type of competition is produced using short range lateral excitatory and inhibitory interconnections with fixed weights (arranged so as to produce a roughly Mexican hat shaped spatial correlation function). Adaptive weights from the input layer are trained with a simple Hebbian learning rule, with a renormalization term to force the weights of each unit to sum to a constant. When trained on simple binary patterns consisting of oriented bars, units learn to respond preferentially to particular orientations. Furthermore, neighboring units tend to respond to similar orientations, while next-to-neighboring units respond to nearly perpendicular orientations, so that the response profile across the lattice of units forms an orientation map qualitatively similar to that seen in visual cortex. Von der Malsburg's learning equation is similar to Oja's simple normalized Hebbian rule, except that they use different norms. The result is very different, however, because the recurrent connections cause units' activities to tend to be positively correlated with their nearest neighbors', and negatively correlated with their more distant neighbors. When this effect is combined with a learning rule which causes units to discover correlations in the input, the result is that adjacent units learn maximally overlapping features, while units separated by short distances discover minimally overlapping features which tend to be *mutually exclusive*.

Fukushima's Cognitron (1975) is a multi-layer version of competitive learning which performs a simple type of hierarchical clustering. The basic model is similar to von der Malsburg's; the main differences are a slightly more complicated multi-layered architecture with probabilistically interconnected units, strictly inhibitory local lateral connections within layers, and an explicit winner-take-all learning rule: a unit only adapts its weights when it is the most strongly active in its local neighborhood. The receptive field radius increases with progressive layers, so that simple features of small spatial regions are learned in lower layers, and higher order features (corresponding to larger spatial regions of the input) are learned by higher layers.

On a very simple digit recognition task, at the fourth layer, most units learn to respond selectively to single digits. In the Neocognitron (Fukushima and Miyake, 1982), the model is extended to achieve shift-invariant pattern recognition, by replicating local feature detectors at multiple spatial positions within each level in the hierarchy. Ambros-Ingerson, Granger and Lynch (1990) have also proposed a competitive learning algorithm for hierarchical clustering, in a model of the rat olfactory cortex.

Rumelhart and Zipser (1985) have studied a simplified version of competitive learning which captures many of the essential features of the above models, but is much easier to analyze. Rather than implementing the winner-take-all mechanism using lateral inhibition, they dispense with the recurrent links and simply use a nonlinear activation function which sets the activity of the winning unit (the one with the greatest total input) to one, and the rest to zero. They use the following learning rule:

$$\Delta w_{ji}{}^{\alpha} = \begin{cases} 0 & \text{if unit } j \text{ loses on pattern } \alpha \\ \varepsilon\left(\frac{y_i^\alpha}{\sum_k y_k^\alpha} - w_{ji}\right) & \text{if unit } j \text{ wins on pattern } \alpha \end{cases}$$

By redistributing some proportion ε of each the winning unit's weights to the weights on its active input lines, this rule maintains the constraint that $\sum_i w_{ji} = 1$, for each unit j. We can think of this learning rule as causing each unit to move its weight vector toward the mean of the cluster of patterns for which that unit responds. Hence, each unit is performing approximate gradient descent in the squared distance between its weight vector and the patterns in its cluster, which is equivalent to the standard k-means clustering algorithm.

Kohonen's model of unsupervised topological map formation (1982; 1988) has much in common with the competitive learning models discussed so far. He uses a network of units arranged in an array (usually two-dimensional) such that there is some topological neighborhood, $N_i(t)$, defined for each unit i. The "winning unit" c is the one for which the Euclidean distance between its weight vector and the current input vector is minimal. Every unit within the winner's neighborhood, N_c, adapts its weights according to the following learning rule:

$$\Delta w_{ji}{}^{\alpha} = \begin{cases} \varepsilon(y_i^\alpha - w_{ji}) & \text{if unit } j \in N_c \\ 0 & \text{otherwise} \end{cases}$$

If both the learning rate ε and the neighborhood size shrink gradually over the course of learning, the units' responses tend to become distributed evenly over the input probability distribution. For neighborhoods of size 1, Kohonen's algorithm is equivalent to k-means clustering (Kohonen, 1988). For larger neighborhoods, the algorithm is a generalization of k-means which adapts each weight toward the centre of its own cluster of patterns *and* its neighbors' clusters, resulting in an ordered mapping that tends to preserve the topological structure of the input distribution. Kohonen has applied this algorithm to preprocessed speech data, and found that the clusters found by units usually correspond to phonemes. The sequences of these "quasi-phonemes" produced by processing a sequence of time slices of the speech signal can be viewed as an ordered trajectory through a "phonological map", indicating that the network has learned to represent similar sounds at nearby locations in the map.

With a more elaborate architecture employing both feed-forward and recurrent feedback connections, Grossberg's version of competitive learning (Grossberg 1987; Carpenter and Grossberg, 1983,1987), allows some degree of sequential matching, so that a new cluster will be formed only if a new pattern is sufficiently different from all of the clusters formed so far. By appropriately tuning the model parameters, one has control over the plasticity-stability tradeoff, so that the system tries to flexibly adapt to new classes of inputs without destroying the representations it has learned so far.

A further extension to the basic competitive learning model is to make each unit's response be a function of the distance between each input pattern and its weight vector, so that a unit responds in proportion to the closeness of each pattern to its weight vector (rather than having a binary response). One possible response function is a Gaussian of the distance; each unit therefore becomes maximally tuned to some particular region of the input space, and this tuning falls off exponentially in all directions. Moody and Darken (1989) applied this type of competitive learning as an unsupervised preprocessing stage to speed up subsequent supervised learning. They trained an adaptive layer of Gaussian units using the standard k-means clustering algorithm to adjust the Gaussian centres; an additional layer of units was then trained by a supervised learning procedure (mean squared error minimization) to solve two difficult problems: phoneme recognition and chaotic time series prediction. They report that this hybrid algorithm results in about two orders of magnitude speedup in the learning time compared to pure supervised learning with back-propagation. The layer of Gaussian units speeds learning for two reasons: First, two input vectors which are far apart (in Euclidean distance) will tend to activate non-overlapping sets of Gaussian units, so there will be no interference between these two training cases. Second, the incoming weights of Gaussian units do not depend on the outgoing weights so there is modularity of learning between these two layers of weights.

Nowlan (1990) proposed a "soft competitive learning" method; rather than only allowing the winner (or winning neighborhood) to adapt, each unit can adapt its weights for every input case, in proportion to how strongly it responds on a given case. As a group the units form a good multi-modal model of the underlying distribution, by performing gradient ascent in the model likelihood. The learning procedure is similar to Dempster, Laird and Rubin's EM algorithm (1977) for the special case of Gaussian mixture components. Nowlan showed that this method is superior to the traditional "hard competitive learning models" on two classification tasks, hand-written digit and vowel recognition (when the unsupervised learning for each algorithm is followed by a linear supervised layer). Furthermore, compared to a nonlinear multilayer Back Propagation network on the digit task, the soft competitive model required roughly an order of magnitude fewer learning iterations to achieve comparable classification performance.

2.2.5 Algorithms that maximize information transmission

Several of the papers discussed so far have viewed the goal of unsupervised learning as finding an optimal encoding of the input patterns. For example, it has been pointed out that PCA is the optimal linear encoding with respect to mean squared reconstruction error (Sanger,1989a,b,c; Cottrell, Munro and Zipser, 1987). .

An alternative optimality criterion proposed by Barlow (1985,1989) is to find an encoding which is minimally *redundant*. Barlow suggests that the goal of unsupervised learning should be to find an encoding of the input which makes it easy to form new associations between sensory events and reward/punishment. If the encoding of the sensory input vector into an n-element feature vector has the property that the n elements are statistically independent, then all that is required to form new associations with some event V (assuming the features are also approximately independent conditioned on V) is knowledge of the conditional probabilities $p(V|y_i)$, for each feature y_i (rather than complete knowledge of the probabilities of events conditional upon each of the 2^n possible sensory inputs). Barlow proposes that one way to achieve featural independence is to find a *minimum entropy encoding*: an invertible code (i.e., one with no information loss) which minimizes the sum of the feature entropies. Although this permits high "within-feature redundancy" (i.e., the expected values of individual features may deviate from 0.5), it minimizes the "between-feature redundancy" (the extent to which one feature is predictable from some combination of the other features). Barlow's minimum entropy objective intuitively seems to capture the desirable characteristics of a sensory processing system; unfortunately he does not propose any direct method of finding such a code. A simpler objective of learning *decorrelated* codes via an anti-Hebbian learning rule has been proposed by Barlow and Földiák (1989). Some serial heuristic search algorithms for finding minimum entropy codes are described by Barlow, Kaushal and Mitchison (1989). Földiák (1990) has shown how a network of units with feed-forward connections trained by a normalized Hebb rule, and lateral feed-back connections trained by an anti-Hebbian rule, can learn a sparse code which approximates Barlow's objective, by reducing the statistical dependency between features while preserving most of the information about the input patterns.

The GMAX algorithm, proposed by Pearlmutter and Hinton (1986), uses an objective similar to Barlow's. The objective is to try to discover features which account for redundancy in the sensory input. The GMAX algorithm causes a unit to discover statistical dependencies between its input lines by maximizing the difference between the output distribution of the unit, P, in response to structured input, and the distribution, Q, that would be expected if the input lines were independent. Using probabilistic binary units, the GMAX algorithm maximizes the asymmetric divergence (introduced in Chapter 1) between these two distributions:

$$G = P \log \frac{P}{Q} + (1 - P) \log \frac{1 - P}{1 - Q}$$

When a unit is trained on images of oriented bars, it learns centre-surround receptive fields much like those learned by Linsker's Hebbian network.

Unfortunately, there is no straightforward generalization of the GMAX principle to multiple output units. Pearlmutter and Hinton propose two possible mechanisms: adding an extra term to the objective function which minimizes the correlation coefficient between the outputs of units, and a mechanism of mutual inhibition. The former would encourage units to learn statistically independent features, whereas the latter would encourage the discovery of mutually exclusive features.

Linsker (1988) has proposed an information theoretic objective function for perceptual processing which he calls the *Infomax* principle: a network should learn a mapping which preserves as much information as possible about the input vector. Linsker analyzes the consequences of this principle for two special cases. In the first case, the output of a unit is a linear function of its total input plus a noise term n. Both the input and noise are assumed to have Gaussian distributions. For this case, the rate at which the unit's output y_j transmits information about its input is:

$$R = 0.5 \log \left(\frac{V(y_j)}{V(n)} \right)$$

where $V(n)$ is the variance of the noise. For this model, assuming the noise variance is fixed, maximizing R is equivalent to maximizing the variance of the unit's output. (Note that this is equivalent to Hebbian learning, as mentioned earlier.)

In the second model Linsker considers, there is Gaussian noise n_i of variance $V(n)$ added to each (Gaussian) input line i. Now the information rate is:

$$R = 0.5 \log \left(\frac{V(y_j)}{V(n) \sum_i {w_i}^2} \right)$$

For this model, the maximum information rate is achieved when the output variance is maximized while the length of the weight vector is minimized. In this case, we have a principal component analyzing unit.

The situation becomes more interesting for models with multiple units. Plumbley and Fallside (1988) analyze the case of a linear network with additive Gaussian noise which performs dimensionality reduction. They note that the information loss in the mapping from the inputs to the outputs is bounded from above by the entropy of the error in reconstructing the inputs. This entropy can be minimized by minimizing the mean squared reconstruction error. So in this case, the optimal encoding with n units is the first n principal components. Linsker (1988) also analyzes this case; he points out that the information rate for a collection of linear units with Gaussian noise is:

$$R = 0.5 \log \left(\frac{Det(Q^y)}{V(n)} \right)$$

where $Det(Q^y)$ is the determinant of the covariance matrix of the output vector y. This results in a tradeoff between maximizing the variances of the outputs, and decorrelating them. If the noise variance is large, the latter term predominates in the calculation of the determinant, and some redundancy is therefore desirable in increasing the signal to noise ratio. In the absence of noise, assuming there is some dimensionality reduction, the optimal solution is a decorrelated set of outputs having maximal variance. Note that Barlow's minimum redundancy principle achieves the same end as Linsker's Infomax principle only in the latter case; as the noise level increases, the Infomax principle becomes one of *maximal* ("between-unit") redundancy.

Atick and Redlich (1990) explore an extension of Barlow's minimum redundancy principle which applies to noisy channels. The model they study is of a perceptual system which receives as input $x = s + n_1$, a noisy (i.e., quantized) version of some underlying signal s. The goal of the perceptual system is to find a mapping into a (slightly noisier) vector, $y = Ax + n_2$ (where A is a linear matrix operator), which minimizes the following redundancy measure:

$$\mathrm{R} = 1 - \frac{I(y;s)}{C(y)}$$

(where $I(y;s) = H(y) + H(s) - H(y,s)$ is the mutual information between y and s, and C is the channel capacity) subject to the constraint of no information loss from x to y:

$$I(y;s) = I(x + n_2; s)$$

It is assumed that $C(y) >= I(x;s)$, i.e. there is no dimensionality reduction in the mapping from x to y. Here, $C(y)$ is taken to be the maximum of $I(y;s)$ when the power in the output signals $\langle y_i^2 \rangle$ is held constant and there is no noise apart from quantization. In this case, if the output signal and noise have Gaussian probability distributions, the channel capacity has the following form:

$$C(y) = 0.5 \log \left(\frac{Det(Diag(Q^y))}{Det(Diag(Q^\delta))} \right)$$

where $Diag(Q^y)$ is the matrix consisting of the diagonal elements of the covariance matrix of y, and zeroes elsewhere, and δ is the quantization noise. To obtain an explicit solution which minimizes the above redundancy measure, they propose using a Lagrange multiplier to implement the constraint of zero information loss, and minimize the following "generalized redundancy measure":

$$\mathrm{R}' = C(y) - \lambda \left[I(y;s) - I(x + n_2; s) \right]$$

In a noise free system, the redundancy can be squeezed to zero by minimizing the diagonal terms of Q^y, thereby lowering the channel capacity (as defined above). This is equivalent to Barlow's minimum redundancy principle (for the special case of a linear system with Gaussian variables), which says that we should minimize the sum of the component entropies of y in order to obtain uncorrelated components. In a noisy system, however, much of the channel capacity is wasted by transmitting noise; the signal to noise ratio is improved in this situation by allowing correlated output components. Bialek, Ruderman and Zee (1991) have analyzed a similar objective function, combining an information capacity term with fixed gain constraints on the filters, and penalties for long-range connections. When they analyzed the behaviour of the system on natural images, they found that the resulting spatial frequency filters are very similar to the response properties of mammalian visual cortical cells and the retinal ganglion cells of lower vertebrates.

The difference between Atick and Redlich's generalized redundancy measure and Linsker's Infomax principle is that when the output has the same dimensionality as the input, the latter would yield infinitely many equivalent solutions; maximizing information transmission (i.e., the determinant of Q^y) can be achieved by any linear transformation $y = Ax$ such that the output spans the same space as the input. On the other hand, redundancy can be lowered in two ways (depending on the level of noise): by increasing $I(y;s)$, and by keeping I constant while decreasing C, thus decorrelating the outputs.

2.3 Discussion

2.3.1 How can we best model the input distribution?

If our goal is to model the input distribution as accurately as possible, then we would like to be able to encode efficiently all the "interesting features" in the data. However, it is unlikely that a single unsupervised learning method can do this in reasonable time for any arbitrary input distribution, without making some *a priori* assumptions about the kinds of structure in the environment. Each of the approaches we have mentioned can be expected to perform well on certain distributions, and poorly on others.

Algorithms related to Principal Components Analysis (PCA) learn a set of linear orthogonal projections in the directions of principal variation in the input distribution, or a rotated subspace of these directions. In some cases, the principal components decomposition may coincide with features of interest in the input. Oja (1989) suggests that subspace methods are particularly useful for representing and classifying patterns such as spectra and histograms.

For arbitrary input distributions, there is no guarantee that a subspace method or PCA will capture the interesting structure. PCA would be expected to represent poorly parameters which vary in a highly nonlinear manner, and in this case, nonlinear autoencoders would be expected to do much better. In data with many isotropically distributed clusters, PCA would in fact tend to obscure clusters in the data (Huber, 1985). In this situation, competitive learning schemes may be more appropriate, since they attempt to represent explicitly the locations of clusters.

Boltzmann machines (including Freund and Haussler's model) and Neal's connectionist belief networks attempt to explicitly model the probability distribution of a collection of patterns by discovering hidden features that explain correlations between the input components. While these methods are potentially very powerful, in attempting to capture *all* of the structure in the data in one stage, they tend to be good at finding low order structure, and less good at finding very high order features which are not easily expressible as combinations of low order features (Radford Neal, personal communication).

2.3.2 What kind of representation is best for later learning?

In designing unsupervised learning algorithms, in addition to the question of what type of representation best models the data, we must also consider what type of representation will be most convenient as input for further learning.

PCA has the nice property of finding a set of *uncorrelated* projections. Converting the input to this representation will typically be helpful for a method such as Back Propagation learning by steepest descent. One thing that makes supervised Back Propagation learning slow is that there may be interacting effects of different weights on the error. Because of these interactions, convergence can be expected to be very slow, even with simple local accelerating procedures such as momentum (Plaut, Nowlan, and Hinton, 1986) or delta-bar-delta (which employs an adaptive learning rate for each weight based on gradient information (Jacobs, 1987)). If, on the other hand, the input representation uses uncorrelated components, then the Hessian of the error function is more diagonally dominant, so simple acceleration methods will permit a considerable speedup, by scaling the learning rates appropriately along each weight axis independently. Of course, this analysis ignores the non-linearities of a multi-layer Back Propagation network, but it may still provide a useful heuristic for designing good input representations even in the nonlinear case.

If later learning requires interpolating a smooth function from points in the input distribution, then cluster centers, or more generally any set of radial basis functions, will make a good set of interpolation points (Moody and Darken, 1989; Renals and Rohwer, 1989; Poggio, 1989). However, in some situations there are disadvantages to this type of representation. We are "spreading out" the input distribution into a higher dimensional space, and effectively partitioning it into a number of (more or less) disjoint sub-regions of feature space, as illustrated in Figure 2.4. In doing so, we may be scattering essential information about the continuity of individual features, and about combinations of these features, across many clusters. Suppose, for example, that there is a weak correlation between the desired output and the first component of the input vector. In a system that uses narrowly-tuned radial basis functions, information about the weak correlation will be spread over many different RBF's and it will be hard to detect the correlation. RBF's make it easier to detect correlations between the desired output and very high order combinations of input values, but harder to detect some of the low order correlations that are easily detected by systems that use a less local representation.

2.3.3 What have we gleaned from neurobiology?

The neurobiological literature suggests that there are certain "universal" coding principles employed at the earliest stages of sensory processing which are ubiquitous in animal perceptual systems, such as local

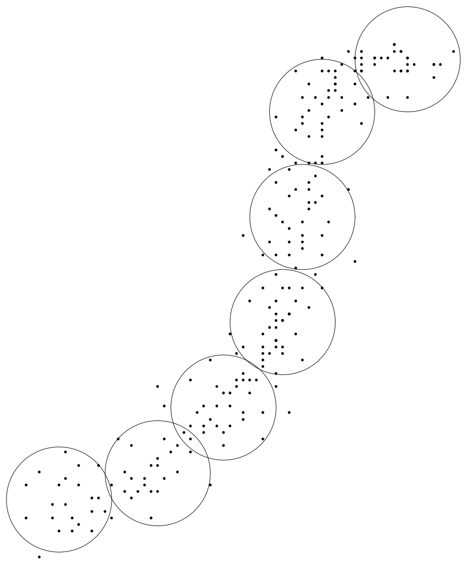

Figure 2.4: *A two-dimensional data distribution, with circles illustrating the locations where a typical clustering algorithm (e.g. k-means or competitive learning) would place the cluster centers.*

smoothing of the signal in space and time. These early signal processing stages are accounted for very well by information transmission models which assume some principal of maximal information preservation subject to hardware limitations (Barlow, 1985; Barlow, 1989; Linsker, 1988; Atick and Redlich, 1990; Bialek, Ruderman and Zee, 1991). But it is probably not fruitful to keep applying the same information transmission principles to higher and higher layers of processing (without any additional constraints), and expect to account for all of perception, as these models are too unconstraining. Beyond the first few layers of processing, there seems to be a transition from universal regularity detectors (such as spatial frequency analyzers) which tend to preserve *all* of the information in the signal, to higher level, environment-specific and/or behaviourally salient feature analyzers.

Information transmission, clustering, and PCA-related algorithms are useful for achieving data compression, noise reduction, and local spatio-temporal smoothing. These are good properties for signal prediction/reconstruction domains, and also for preprocessing sensory information for subsequent interpretation. However, it appears that brains do much more than simply try to reconstruct the signals sampled by their sensors, but rather, some underlying "meaningful variables" or features are extracted which allow high level interpretations. So two obvious question are: What is the next step in modeling perception? And is it possible to apply unsupervised learning principles any further? To answer these questions, we must search for more constraining objectives for perceptual learning which will force the network to build models of perceptually relevant structure in the world. One approach is to make constraining assumptions about the kind of structure we are looking for and wire it into the network's architecture and/or objective function. This thesis explores such an approach.

Our major goal is to build self-organizing network modules which capture important regularities of the environment in a simple form suitable for further perceptual processing. We would like an unsupervised learning procedure to be applicable at successive layers, so that it can extract and explicitly represent progressively higher order features. If at one stage the algorithm could learn to explicitly represent continuous real-valued parameters such as relative depth, position and orientation of features in an image, subsequent learning could then discover higher order relations between these features, representing, for example, the location of object boundaries.

Chapter 3

Coherence-based unsupervised learning: Discrete Imax

Previously proposed objectives for unsupervised learning (e.g., minimizing reconstruction error or information loss) have attempted to encode *all* the information in the data set, while making minimal assumptions about the kind of structure in the data. In the previous chapter, we reviewed many examples of this approach, and discussed their limitations. In this dissertation, we take an alternative approach, which is to attempt to constrain the learning problem by restricting the features of interest, in some way, to those useful for further perceptual processing. In this way, we hope to extract higher order features that will be useful for robust signal interpretation. Further, this should provide a way of speeding up difficult learning problems by decomposing them into several unsupervised feature-extraction stages (which could potentially be followed by a simpler supervised stage).

Our first task is to specify what sort of structure an unsupervised learning procedure should try to discover. We would like it to find features that are sufficiently general that they will likely occur in a variety of situations, and further, that are salient for perceptual processing. One kind of structure that is ubiquitous in sensory information is spatio-temporal coherence. By "coherence" we simply mean that one part of the signal can be somehow predicted from another part. The prediction may be based on an equality relation (e.g., the signal is equal for spatially or temporally adjacent samples), a linear relation, or any higher order relation.

For example, in speech signals, speaker characteristics such as the fundamental frequency are relatively constant over time. At shorter time scales, individual words are typically composed of long intervals having relatively constant spectral characteristics, corresponding to vowels, with short intervening bursts and rapid transitions corresponding to consonants. Even the consonants change across time in very regular ways. This temporal coherence at various scales makes speech predictable, to a certain degree. Similarly, visual, olfactory and tactile sensations exhibit coherence across space and time. In the visual domain, adjacent parts of the same object stimulate nearby photoreceptors in the retina. Nearby regions of the same object are usually coherent with respect to many parameters, such as texture, orientation, colour and depth; thus nearby photoreceptors tend to sample spatially coherent signals. Since most objects in the visual world move slowly, if at all, the visual scene changes slowly over time, exhibiting the same temporal coherence as other sensory sources. Further, there is coherence *across* sensory modalities. When one examines an object, both visual and tactile cues can provide consistent information about features such as the object's texture, hardness, orientation in space, and even the identity of the object. When one listens to a speaker, the auditory signal may at times be unintelligible by itself, but the visual signal (the shape of the speaker's mouth, etc.) provides disambiguating information as to which word was spoken. Thus, it seems that spatio-temporal and multi-sensory coherence provide important cues for segmenting signals in space and time, and for object localization and identification.

3.1 Maximizing mutual information between outputs

We have proposed that a good objective for unsupervised learning is to extract higher-order features from the sensory input that exhibit simple forms of coherence across time or space (Becker and Hinton, 1989). If we sample different parts of an image, for example, we find that there is much redundancy; spatially nearby intensity values tend to have high mutual information. However, this information is in a rather complex form: a variety of parameters, such as the surface orientation, depth, and reflectance properties, are encoded in the observed pixel intensities. We would like to transform the raw sensory input so that the mutual information between spatially nearby regions can be expressed in a simpler form, by explicitly representing the underlying parameters of interest.

We can extract a spatially coherent image feature using a connectionist architecture like the one shown in figure 3.1. A separate network module is assigned to each image patch. Each module computes an output which is a nonlinear transformation of the image intensities in that patch, representing some image parameter.

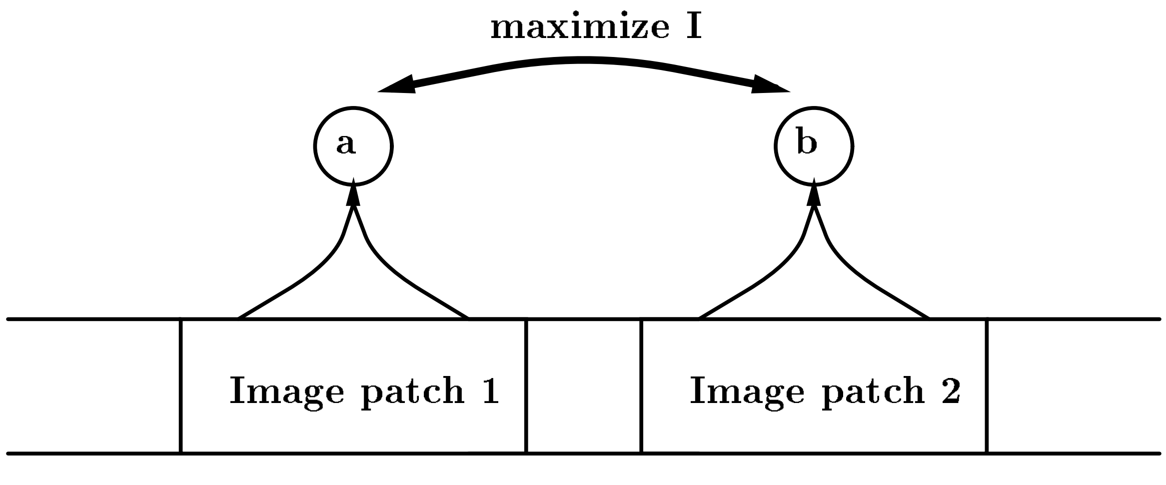

Figure 3.1: *Two units receive input from adjacent, non-overlapping parts of the image. The goal of the learning is to maximize the mutual information between the units' outputs.*

One way to get two neighboring modules to discover a feature that is the same in neighboring image patches is to simply minimize the squared error between their outputs. However, the modules could trivially minimize the squared error by always producing the same outputs for every input pattern. So we need an objective function that measures not only how well the outputs agree, but also whether they are detecting an interesting feature of the image.

A good measure of interestingness of a feature is the Shannon information content of the feature. If the feature extracted from an image patch conveys a high degree of information about the image, it has captured a significant aspect of the structure in the image. However, this criterion alone is not sufficient to force features extracted from neighboring patches to have high agreement.

We need a measure which captures the notions of both good agreement between features, and high information content of individual features. A measure which satisfies both criteria is the mutual information between the two features (defined in Chapter 1). If two neighboring modules produce outputs a and b, their mutual information is given by:

$$I_{a;b} = H(a) + H(b) - H(a,b) \tag{3.1}$$

where $H(a) = -\langle \log p(a) \rangle$ is the entropy of a, and $H(a,b) = -\langle \log p(a,b) \rangle$ is the entropy of the joint distribution of a and b. During learning, the weights in the network are adjusted over many iterations through an ensemble of patterns, to maximize the mutual information between the outputs of neighboring modules.

The choice of this mutual information objective function was inspired by the work of Peter Brown, Robert Mercer and colleagues on modelling the statistical regularities of text (Bahl et al., 1989; Brown et al., 1990b). They developed an unsupervised word classifier, based on the assumption that a word can be predicted more accurately from previous ones if the previous words can be grouped into equivalence classes. Their algorithm used the mutual information between the word class and the next word to be predicted as a criterion for discovering good predictor variables (word classes). They have subsequently applied these methods to related speech recognition problems, including word sense disambiguation (Brown et al., 1991) and machine translation (Brown et al., 1990a).

For the simple case of two binary probabilistic units, we can estimate the mutual information by sampling their activity values over a large set of input cases. Once we have estimated the probabilities of each unit being on and the pair being on together, we can directly compute their mutual information. Similarly, we can directly compute the derivative of their mutual information with respect to these probabilities (and hence, with respect to the weights), as shown in appendix A. The final expression we use for the partial derivative of the mutual information between the outputs of the ith and jth units with respect to the expected output of the ith unit on training case α, p_i^α, is:

$$\frac{\partial I_{y_i;y_j}}{\partial p_i^\alpha} = -P^\alpha \left[\log \frac{p_i}{p_{\bar{i}}} - p_j^\alpha \log \frac{p_{ij}}{p_{\bar{i}j}} - p_{\bar{j}}^\alpha \log \frac{p_{i\bar{j}}}{p_{\bar{i}\bar{j}}} \right] \tag{3.2}$$

where P^α is the probability of training case α, $p_i = < y_i >$ is the expected value of the ith unit's output averaged over the fluctuations for each training case and also over the whole ensemble of cases (when y_i is treated as a stochastic binary variable), $p_{\bar{i}} = <(1 - y_i)>$, and $p_{ij} = < y_i y_j >$, etc.

3.2 An easy learning problem

A simple pattern classification problem is to discriminate sinusoidal intensity patterns of different spatial frequencies and phases, like those shown in figure 3.2. The problem is simple in the sense that a linear filter can be designed which will respond optimally to a particular spatial frequency at a particular phase. Thus, the classification problem is linearly separable (i.e., the problem of detecting *phase-specific* frequencies), and could be solved by a neural network without hidden units. Since each pattern contains only a single unvarying spatial frequency ... nt and should be learnable by the Imax algorithm.

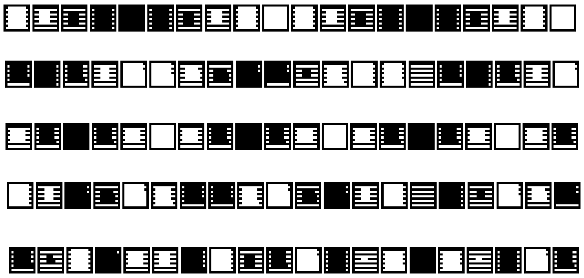

Figure 3.2: *Real-valued input patterns with different spatial frequencies and phases.*

We trained a pair of units on patterns like those shown in figure 3.2, using the architecture shown in figure 3.1. Each unit received input from 10 adjacent pixels of the one-dimensional images, and there was a 2 pixel gap between the receptive fields of the two units. Without this gap, units tended to learn that the immediately adjacent pixels at the edges of their receptive fields have correlated intensities, and to ignore spatial frequency and phase. We used four different training sets, with varying ranges of spatial frequencies:

1. 200 training patterns with spatial frequencies of .2 and .3 cycles per pixel, and 100 examples of each spatial frequency at different phases.
2. 400 training patterns with spatial frequencies of .1, .2, .3 and .4 cycles per pixel, with 100 examples of each spatial frequency at different phases.
3. 900 training patterns with 30 spatial frequencies between .2 and .3 cycles per pixel, and 30 examples of each spatial frequency at different phases.
4. 900 training patterns with 30 spatial frequencies between .1 and .4 cycles per pixel, and 30 examples of each spatial frequency at different phases.

In all cases, phase varied from 0 to 2π.

Each stochastic binary unit used the logistic nonlinearity $f(x) = 1/(1+e^{-x})$ to determine the probability of outputting a 1 as a function of its total weighted summed input x. Rather than running stochastic simulations of probabilistic binary units, we actually used a deterministic, exact algorithm, in which the output of each unit on a particular case α is taken to be its expected value $f(x^\alpha)$. In practice this yields equivalent behaviour, but is much faster since we can obtain good estimates of the various probability statistics required for our learning algorithm in only one sweep through the training set. Each learning iteration required two sweeps through the batch of training patterns, one to compute the probability statistics, and one to compute the mutual information gradients with respect to the weights. Weights were updated using steepest ascent with a fixed step size of 0.1, i.e., on each iteration, each weight w_{ji} was incremented by $-0.1\frac{\partial I}{\partial w_{ji}}$. Five learning runs were simulated, from different initial random weights, for each of the four training sets described above. Each run consisted of 300 learning iterations on the entire training set, except the fourth pattern set which required 800 iterations. The learning usually converged after 100-200 iterations, except in the last condition.

The mutual information after learning, averaged over five learning runs, for the four different training pattern sets listed above, was as follows: 1) .88, 2) .88, 3) .76, and 4) .32 bits. In all cases, units became highly tuned to spatial frequency and phase, as shown in figure 3.3. The first two training sets, containing only a small number (two and four) of spatial frequencies, were the easiest to learn. In these cases, units were able to achieve nearly the maximum possible mutual information by dividing up the phase-frequency space into two equal-sized regions, and achieving near-perfect agreement on each case. The weights learned for the first training set are shown in figure 3.4. When the training set contained many spatial frequencies, the learning problem became more difficult, and units typically learned to divide up the frequency-phase space into a number of disjoint regions.

To achieve high mutual information, a pair of units must have high individual entropies as well as low joint entropy. Maximal individual entropy is achieved when a unit is on with probability 0.5, i.e., when the unit's probability density is evenly distributed over the probability space. Minimal joint entropy is achieved when all the probability density in the joint distribution is concentrated in a small region of the space. Taken together, these two factors drive the units to try to be in agreement on every case (or to have opposite values on every case), and to each be strongly on half the time and strongly off the other half. This is a reasonable representation for spatial frequency in a two-way classification problem, but is inadequate when there are many spatial frequencies. In the next subsection, we address the problem of representing multi-valued spatially coherent parameters of the data.

3.2.1 Imax between n-valued variables

One way to extend Imax to handle multi-valued spatially coherent features, is to maximize the mutual information between two discrete n-valued variables rather than binary variables. A set of n units can be forced to represent a probability distribution over the n states of a discrete random variable $A \in \{a_1 \cdots a_n\}$, by adopting states whose probabilities sum to one. This can be done, for example, by using the "softmax" activation function suggested by Bridle (1990):

$$P(A = a_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{3.3}$$

where x_i is the total weighted summed input to the ith unit.

The mutual information between two n-valued variables, A and B, can be computed in a straightforward manner, once we know the probabilities of each value of A and B, as well as all the pairwise probabilities:

$$I_{A;B} = -\sum_i P(A = a_i) \log P(A = a_i) - \sum_j P(B = b_j) \log P(B = b_j) \tag{3.4}$$

$$+\sum_{ij} P(A = a_i, B = b_j) \log P(A = a_i, B = b_j) \tag{3.5}$$

$I_{A;B}$ can be differentiated with respect to each weight, as shown in appendix A. The derivative of I with

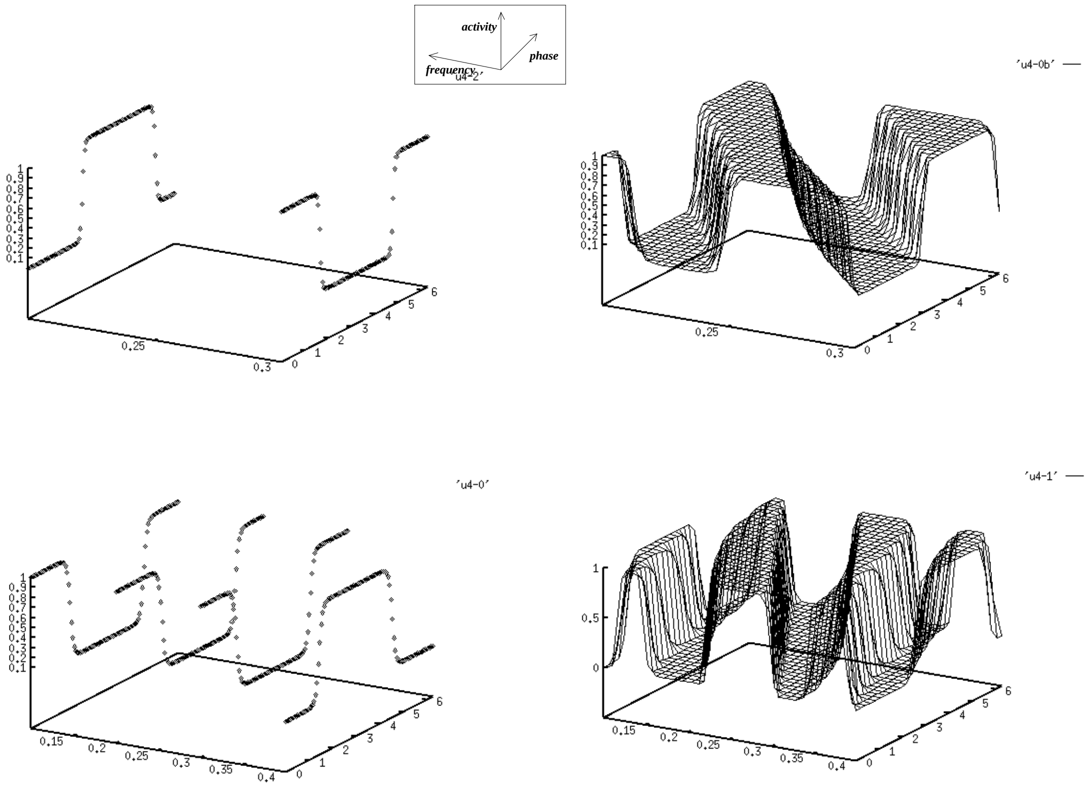

Figure 3.3: *Typical responses of single units versus spatial frequency and phase, over the entire pattern ensemble, for four different training sets. For the top two runs, spatial frequency varied from 2 to 3 cycles per 10 pixels (10 pixels is the width of a unit's receptive field), and for the bottom two it varied from 1 to 4 cycles per 10 pixels. For the left two runs there were only a small number of spatial frequencies (two for the top left, four for the bottom left), and for the right two runs spatial frequency varied (relatively) continuously between the two extremes. In all cases, phase varied continuously from 0 to 2π.*

respect to the total input to the ith unit on case α is:

$$\begin{aligned}
\frac{\partial I_{A;B}}{\partial x_i^\alpha} &= -P^\alpha \sum_j \frac{\partial P(A=a_j|\alpha)}{\partial x_i^\alpha} \left[\begin{array}{l} \log P(A=a_j) - \\ \sum_k P(B=b_k|\alpha) \log P(A=a_j, B=b_k) \end{array} \right] \\
\frac{\partial P(A=a_j|\alpha)}{\partial x_i^\alpha} &= \begin{cases} P(A=a_i|\alpha)(1-P(A=a_i|\alpha)) & \text{if } i=j \\ -P(A=a_j|\alpha)P(A=a_i|\alpha) & \text{otherwise} \end{cases}
\end{aligned} \tag{3.6}$$

where P^α is the probability of training case α.

This learning procedure was simulated on a network consisting of two groups of 12 units, representing two 12-valued variables. The probability of each unit turning on was determined by the softmax equation 3.3. Each group received input from one half of the 10-pixel images. The input patterns to the network were the same as the most difficult training set described in the previous section: a set of 900 patterns with 30 spatial frequencies varying from 0.1 to 0.4 cycles per pixel, and 30 phases varying from 0 to 2π. The network was trained, as before, using steepest ascent with a fixed step size of 0.1, for 300 iterations. Five

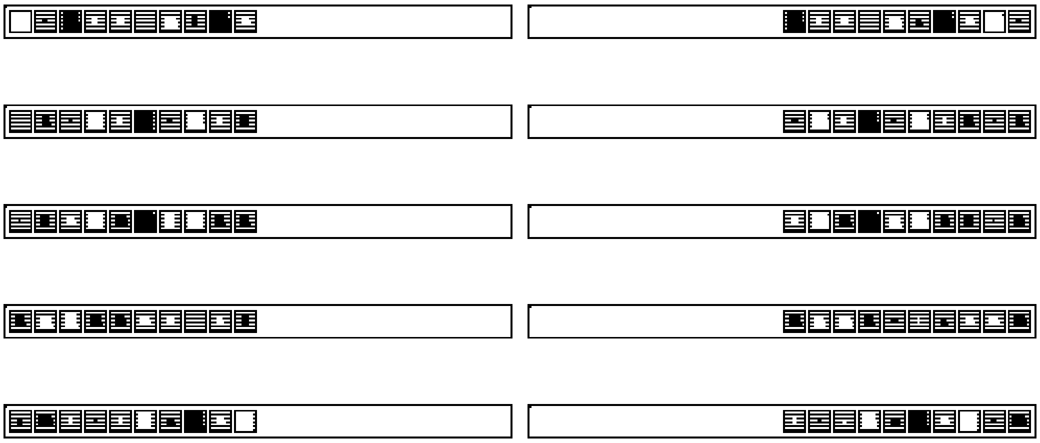

Figure 3.4: *The weights learned by a pair of information-maximizing units trained on patterns containing spatial frequencies of two and three cycles per 10 pixels. The five rows represent the weights learned in five runs, starting from different initial random weights.*

runs starting from different initial random weights were simulated.

Figure 3.5 shows the responses of one group of 12 units after training on one of the runs, and figure 3.6 shows the weights learned by these units on the same run. The network learned to divide up the frequency-phase space fairly evenly among 11 of the 12 units. Each of these 11 units responds to a single continuous region of space, apart from one small glitch in the response profile of the first unit, shown in the upper left corner. Note that phase is periodic, and the scale varies from 0 to 2π, so the end points of the phase scale are equal. In the five runs, the network learned very similar solutions, always dividing up the space between 9 to 11 of the units. The mean mutual information between the two groups, averaged over the five runs, is 1.67 bits. The maximum possible mutual information between two 12-valued variables is $-\log \frac{1}{12} = 3.58$ bits.

The reason that the spatial-frequency/phase classification problem is easy is that it is possible for a single unit to learn a set of weights that perfectly matches a pattern of a particularly frequency and phase. This unit will also respond well to patterns nearby in *both* frequency and phase. The problem would be much more difficult if units had to respond to all instances of a particular spatial frequency independently of phase, or vice versa. The problem of detecting a pattern of a particular frequency at different phases falls within the class of problems shown by Minsky and Papert (1987) to be not linearly separable.

3.3 A more difficult learning problem

A more difficult problem for which the binary version of Imax works well is the task of discovering depth in simple binary stereo images. The stereo or shift-detection problem is difficult because it amounts to detecting translation-invariant features. [1] Jepson and Jenkin (1989) have shown that relative disparity

[1] In this respect, the shift detection problem is very much like the spatial frequency detection problem (i.e., detecting frequency independent of phase). In fact, the two problems are closely related; both involve detecting the distance between corresponding intensity peaks. In the stereo problem it is the disparity between peaks in the left and right images that is important, while in the spatial frequency problem it is the spacing of peaks across the entire image. The random dot stereogram problem is made slightly more difficult by the fact that intensity peaks occur at random intervals.

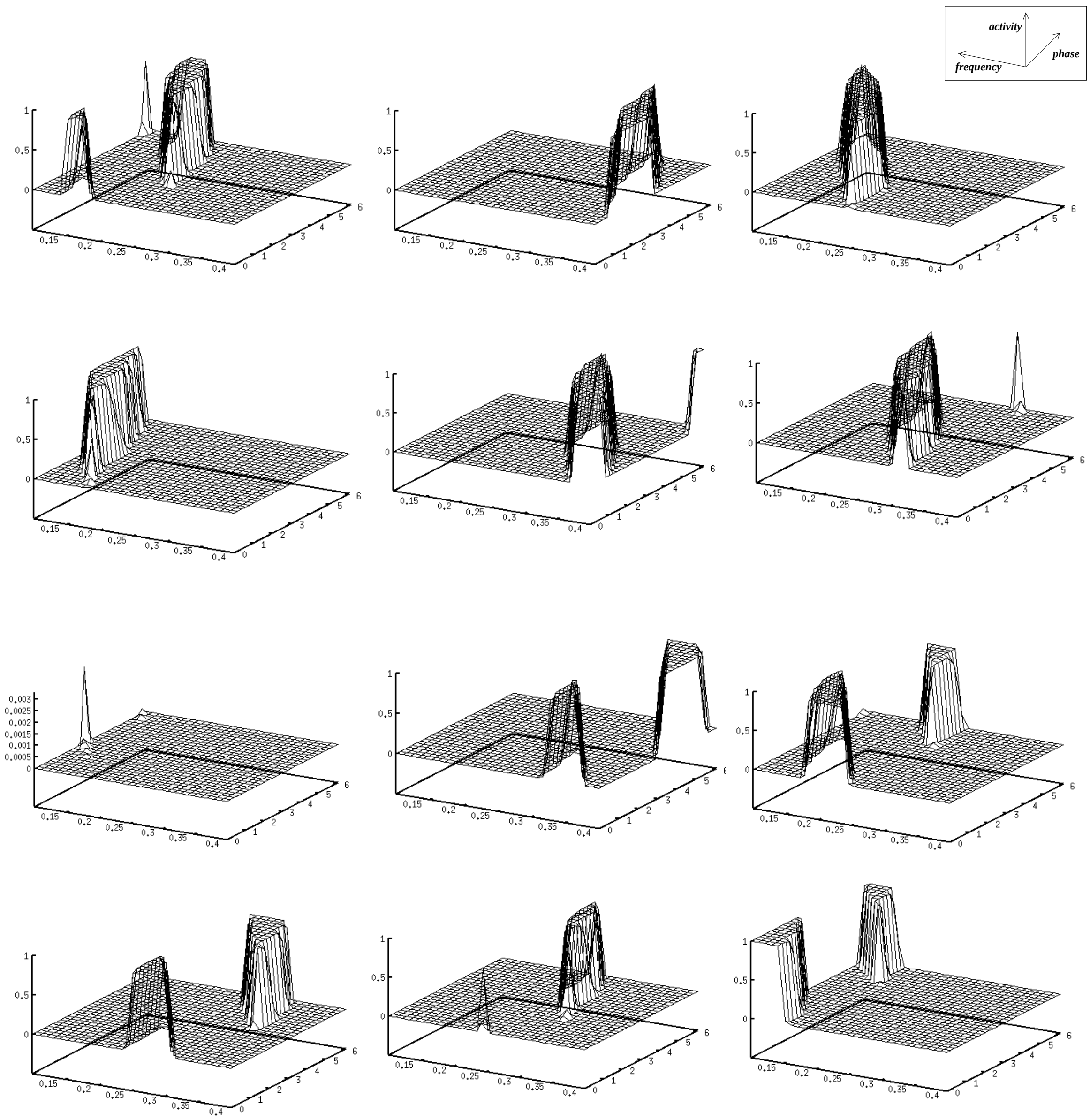

Figure 3.5: *Responses of 12 units versus spatial frequency and phase, over the ensemble of 900 training patterns. The 12 units were trained to maximize mutual information with 12 neighboring units, using a 12-valued discrete code. Spatial frequency in the training ensemble varied from 1 to 4 cycles per 10 pixels, and phase varied from 0 to* 2π. *30 spatial frequencies and 30 phases were used.*

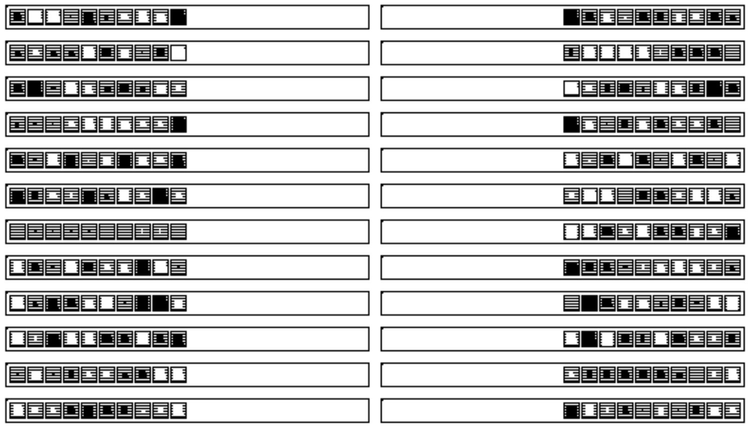

Figure 3.6: *Weights learned by two sets of 12 units trained to maximize mutual information using a 12-valued discrete code. The left column shows weights for the first set of units, all of which receive input from the left half of the image, and the right column shows weights for the second set, which receive input from the right half. Note that the pairing of weights in the left and right columns in this display is arbitrary; the network is free to try to make any combination of units in the two groups agree.*

within a local stereo image patch can be obtained computationally from the responses of two "sine" and "cosine" Gabor filters (spatially localized band-pass filters) which are sharply tuned to the same frequency, but ninety degrees out of phase. The filters are applied to corresponding left and right image patches. From the four filter responses, R_{sine}, R_{cosine}, L_{sine}, and L_{cosine}, the local difference in phase angle (which is directly related to the disparity) between the (one-dimensional) image signal in the left and right views at a particular spatial location can be approximately computed:

$$\phi_{left} - \phi_{right} \simeq \tan^{-1}\left(\frac{R_{sine}L_{cosine} - L_{sine}R_{cosine}}{R_{cosine}L_{cosine} + L_{sine}R_{sine}}\right) \tag{3.7}$$

Although the disparity can be obtained as a simple geometric function of a set of linear filter responses, it is clearly not an easy function for a neural network to compute (assuming units use the usual sigmoidal nonlinearity), as it involves several multiplies and a division operation. (The $\tan^{-1}$ transformation is easy, as it is very close to the sigmoidal activation function.)

For our experiments, we used an ensemble of very simple, binary random-dot stereograms, such as those shown in the input layer in Figure 3.7a. Each input vector consists of a one-dimensional strip from the right image and the corresponding strip from the left image. The right image is purely random and the left image is generated from it by choosing, at random, a single global shift. So the input can be interpreted as an approximation to a one-dimensional stereogram of a fronto-parallel surface at an integer depth. The only local property that is invariant across space is the depth (i.e. the shift). Hence, if one unit looks at one area of the two images, and another unit looks at another area, the only way they can provide mutual information about each other's outputs is by representing the depth.

The input patterns consisted of binary 2 by m bit vectors, the left half (bits 1 to m) being random, and the right half (bits $m+1$ to $2m$) being a shifted version of the left half, as shown in figure 3.7. We used two global shifts, one pixel rightward or one pixel leftward. Each 2 by m bit input pattern was divided into n