Journal of Applied Intelligence, 6, 185–205 (1996)

Unsupervised Neural Network Learning Procedures For Feature Extraction and Classification*

SUZANNA BECKER
Department of Psychology, McMaster University, Hamilton, Ont. Canada L8S 4K1

MARK PLUMBLEY
Department of Computer Science, King's College, London, Strand, London WC2R 2LS UK

Editors: F. Pineda

Abstract.
In this article, we review unsupervised neural network learning procedures which can be applied to the task of preprocessing raw data to extract useful features for subsequent classification. The learning algorithms reviewed here are grouped into three sections: information-preserving methods, density estimation methods, and feature extraction methods. Each of these major sections concludes with a discussion of successful applications of the methods to real-world problems.

Keywords: Unsupervised learning, self-organization, information theory, feature extraction, signal processing

1. Introduction

One of the more difficult and important parts of the classification process is the preprocessing of raw data to extract useful and appropriate features. The raw data are often too large or complex to be used directly as input to a classifier, leading to the 'curse of dimensionality' and other generalization problems if insufficient training examples are available. Even when this is not the case, simply reducing the number of variables representing the data can make learning easier in a later classifying stage. In some cases, prior knowledge about the problem can be used to determine heuristic features such as edges in an image classification problem. However, this is not possible in all cases, and may not be the complete solution in cases where such features are identifiable.

In this article, we review unsupervised neural network learning procedures which can be applied to this preprocessing task. In contrast to either *supervised* or *reinforcement* learning procedures, these unsupervised algorithms use no knowledge of the eventual targets or errors of the classification process. They learn their eventual operation purely from observing the raw input data.

The learning procedures reviewed here are grouped into three main categories. In section 2, we consider information-preserving methods. Here we cover principal component analysis (PCA) and principal subspace methods, temporal prediction of information, and independent component analysis (INCA), as well as direct minimization of information loss. In section 3 we consider density estimation methods related to maximum likelihood parameter estimation. These include so-called "one-of-n" encodings, which may be learned using various forms of competitive learning, and more advanced combinatorial representations. Finally, in section 4 we consider methods which aim to extract higher-order features from data, beyond the pre-processing stages in a system. These include the extraction of max-

* The first author is supported by research grants from the James S. McDonnell Foundation (grant #93-95) and the Natural Sciences and Engineering Research Council of Canada. For part of this work, the second author was supported by a Temporary Lectureship from the Academic Initiative of the University of London, and by a grant (GR/J38987) from the Science and Engineering Research Council (SERC) of the UK.

imally "selective" projections, the use of GMAX to detect statistical dependencies in the data, extraction of invariant features, and extraction of spatially invariant features using Imax. Each of these sections concludes with pointers to examples of applications for the techniques discussed within the section.

2. Information preserving algorithms

In this section we consider learning systems which attempt to preserve as much of the information in the input data as possible, while performing some simplification of the data such as dimension reduction. Since these unsupervised learning systems have no access to eventual classification performance, it is difficult to do anything other than attempt to preserve *all* the information in the input. The major part of this section considers principal component analysis (PCA) methods, from single component algorithms through to variants which extract M components.

2.1. *Principal component analysis and subspace methods*

Principal component analysis (PCA) is a popular dimension reduction and feature extraction method. It appears in the literature under various guises, including Factor Analysis [77], the discrete Karhunen-Loéve transform (KLT) [26], and the Hotelling Transform [28]. This variety of names largely reflects its different applications.

Suppose that we have a zero-mean random vector $\mathbf{x} = [x_1, \ldots, x_N]$ distributed with joint proability density $p(\mathbf{x})$ and covariance matrix

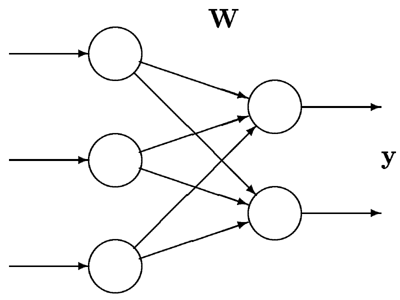

Fig. 1. Three-input two-output PCA network with input $\mathbf{x}$, weight matrix $\mathbf{W}$, and output $\mathbf{y} = \mathbf{W}\mathbf{x}$.

$$\mathbf{Q}^{\mathbf{x}} = E(\mathbf{x}\mathbf{x}^T). \tag{1}$$

The normalized eigenvectors of $\mathbf{Q}^{\mathbf{x}}$ are denoted $\mathbf{e}_1, \ldots, \mathbf{e}_N$, with corresponding eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$. If we were to form a transformed random vector

$$\mathbf{y} = \mathbf{U}\mathbf{x} \tag{2}$$

with the successive rows of an $N \times N$ matrix $\mathbf{U}$ set to the successive eigenvectors $\mathbf{e}_i$ of $\mathbf{Q}^{\mathbf{x}}$, the new random vector $\mathbf{y} = [y_1, \ldots, y_N]$ will have components y_j which are decorrelated from each other, and in order of decreasing variance

$$\sigma_{y_i}^2 = E(y_i^2) = \lambda_i. \tag{3}$$

The first output component y_1 is called the *principal component* of the input, and for any M, the first M output components $[y_1, \ldots, y_M]$ are called the *M principal components* of the input. If this transform is performed by a neural network or similar process, we refer to $\mathbf{x}$ as the *input* vector, and $\mathbf{y}$ as the *output* vector.

In a practical application, instead of a random vector we typically see a sequence of sample vectors $\mathbf{x}(t)$ for $t = 1, 2, \ldots$ distributed according to $p(\mathbf{x})$. Often only a finite amount of data is available, so we only see a finite sequence of τ samples (so $t = 1, 2, \ldots, \tau$). For large τ, the t-average

$$\mathbf{Q}^{\mathbf{x}(t)} = \langle \mathbf{x}(t)\mathbf{x}(t)^T \rangle \tag{4}$$

is a reasonable estimate of the underlying covariance matrix $\mathbf{Q}^{\mathbf{x}}$. We can therefore form an estimate of the principal components of $\mathbf{x}$ from our observations of the sample sequence $\mathbf{x}(t)$. For the remainder of this section, we shall assume that the number of samples is large enough that $\mathbf{Q}^{\mathbf{x}}$ and $\mathbf{Q}^{\mathbf{x}(t)}$ are approximately equal.

In an N-input M-output linear neural network (Fig. 1), PCA is realized simply by setting the weight matrix $\mathbf{W}$ such that the ith row $\mathbf{w}_i$ is equal to the ith eigenvector $\mathbf{e}_i$ of the input covariance matrix. The outputs y_i of the network are therefore the principal components of the input data.

The reason that PCA is useful for preprocessing is because of its information preserving properties. Given a number of outputs M, it is the linear transform which minimizes the mean squared reconstruction error of the input sequence $\mathbf{x}(t)$ from the output sequence $\mathbf{y}(t)$ [26]. Also, it is the linear transform which preserves the maximum amount of Shannon information at the output, under assumptions of uncorrelated equal-variance additive Gaussian noise on the input signal [44], [65]. PCA is therefore an optimal linear dimension-reduction method.

Classical Techniques for PCA If the entire data sample sequence $\mathbf{x}(t)$ for $t = 1, \ldots, \tau$ is available at once, the required weight vectors for PCA can be calculated using standard techniques such as singular value decomposition (SVD) [12]. A sequential version is also available, allowing the SVD to be updated as the data arrives. However, while SVD is an exact method, which will use all of the data in the input data sequence seen so far, it is rather complex for a neural network algorithm. In addition, it finds all N principal components of the input, even if only some small number $M \ll N$ of principal components were needed.

If only the first principal component is needed, this can be estimated by repeated multiplication of an initially random vector by the input covariance matrix $\mathbf{Q}^{\mathbf{x}}$[27]. If we start with a random column vector $\mathbf{w}(0)$ and repeatedly apply the multiplication

$$\mathbf{w}(k+1) = \mathbf{Q}^{\mathbf{x}}\mathbf{w}(k) \qquad (5)$$

the component of $\mathbf{w}(k)$ in the direction of the principal component will soon come to dominate, since it goes up with λ_1^k which quickly becomes much larger than the corresponding factor for all other eigenvectors. Of course, the vector $\mathbf{w}(k)$ will periodically need to be re-scaled to prevent overflow, possibly by renormalizing so that its length $|\mathbf{w}(k)|$ is set to 1 after each step. Once the first principal component has been found, the second can be extracted by subtracting the effect of the first from the input data, and re-applying the modified data to a new random vector.

Unfortunately, while this method is simple enough to be a neural network algorithm, it can only be used when all the input data is available before the procedure starts, so that $\mathbf{Q}^{\mathbf{x}}$ can be calculated. However, many neural network algorithms have been proposed which can find principal components, using relatively simple "online" algorithms operating on the data vectors as they arrive.

Finding the first principal component For our input vector $\mathbf{x} = [x_1, \ldots, x_N]$ and a single output

$$y(t) = \mathbf{w}(t).\mathbf{x}(t) = \sum_{i=1}^{N} w_i(t)x_i(t) \qquad (6)$$

the simple Hebbian algorithm

$$w_i(t+1) = w_i(t) + \epsilon x_i(t)y(t) \qquad (7)$$

which we can write in vector notation as

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \epsilon\mathbf{x}(t)y(t) \qquad (8)$$

will tend towards a vector in the direction of the principal component, but its length will tend to be unbounded. The factor ϵ is a small update factor or 'learning rate' term. For theoretical convergence of algorithms such as those considered below, the update factor should decrease slowly with time, typically as $\epsilon(t) = 1/t$. However, in a practical application, this can often lead to very slow convergence beyond small values of t, so ϵ is often set to a small constant instead. In this case, the algorithm will not fully converge to its stable point, but will make small random movements around it, depending on the size of ϵ.

Oja [54] modified this algorithm so that $\mathbf{w}$ is renormalized after each step, leading to the two-step algorithm

$$\begin{aligned} \tilde{\mathbf{w}}(t+1) &= \mathbf{w}(t) + \epsilon\mathbf{x}(t)y(t) & (9) \\ \mathbf{w}(t+1) &= \tilde{\mathbf{w}}(t+1)/|\tilde{\mathbf{w}}(t+1)| & (10) \end{aligned}$$

which ensures that $\mathbf{w}(t)$ has unit length for all time $t > 0$. The weight vector $\mathbf{w}(t)$ in this algorithm asymptotically converges to the principal component $\mathbf{e}_1$ (or $-\mathbf{e}_1$).

If the update factor ϵ is sufficiently small, Oja [54] also showed that the two-step algorithm above can be approximated by the single-step algorithm

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \epsilon\left(\mathbf{x}(t)y(t) - \mathbf{w}(t)y(t)^2\right) \qquad (11)$$

which is often referred to as "Oja's Rule". The weight vector in this algorithm also converges to the unit-length principal component, but without explicitly normalizing at each step. The '$-\mathbf{w}y^2$' term on the right hand side of (11) tends to decrease the length of $\mathbf{w}$ if it gets too large, while allowing it to increase if it gets to small.

Note in passing that the Oja Rule (11) is also a *local* algorithm, which can be seen more clearly if we write it in the form

$$\begin{aligned} w_i(t+1) &= w_i(t) \\ &\quad +\epsilon\left(x_i(t)y(t) - w_i(t)y(t)^2\right). \end{aligned} \tag{12}$$

The term *local* means that the change to the weight w_i is determined purely by the activations y and x_i at either end of the weight, together with the weight value w_i itself. Most of the neural network algorithms which extract more than one principal component are *not* local algorithms. This is somewhat important if we are looking for an algorithm which is *biologically plausible*, but is not so great a consideration for most practical applications.

Re-estimation algorithms Many algorithms have been suggested which extend the Oja Rule (11) to M output units $\mathbf{y} = [y_1, \ldots, y_M]$ where

$$y_j(t) = \mathbf{w}_j(t)\mathbf{x}(t) = \sum_{i=1}^{N} w_{ji}(t)x_i(t). \tag{13}$$

Many of these work in a hierarchical manner, arranging for the weights to the first output unit to be updated using the Oja Rule, but with the effect of earlier outputs either subtracted or decorrelated away from later units, thus forcing later units to extract different components.

For example, the Sanger [69] Generalized Hebbian Algorithm (GHA) incorporates a form of Gram-Schmidt orthogonalization (GSO) to give the algorithm

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \epsilon y_j(t)(\mathbf{x}(t) - \hat{\mathbf{x}}_j(t)) \tag{14}$$

where $\hat{\mathbf{x}}_j(t)$ is given by

$$\hat{\mathbf{x}}_j(t) = \sum_{k=1}^{j} \mathbf{w}_k(t)y_k(t). \tag{15}$$

An alternative algorithm, also derived from GSO, is Oja and Karhunen's Stochastic Gradient Algorithm (SGA) [57], [56]. This is the same form as GHA, but with (15) replaced by

$$\hat{\mathbf{x}}_j(t) = \mathbf{w}_j(t)y_j + 2\sum_{k=1}^{j-1} \mathbf{w}_k(t)y_k(t). \tag{16}$$

The GHA algorithm (14), (15) and SGA algorithm (14), (16) both force successive outputs to learn later principal components by subtracting estimates of the earlier components from the input before the data reaches the learning algorithm, although the outputs are still calculated from the original input data. They both reduce to the Oja rule (11) for a network with single output. Abbas and Fahmy [1] suggested a related approach. After each successive output unit learns its principal component, the effect of that component is removed by subtracting it from the original data. Using the Oja Rule on the modified input data will find the next principal component.

One possible disadvantage with non-symmetrical algorithms is that they force particular roles on particular units. Although the learning proceeds in parallel, in some sense the first unit has to find the first principal component before the second can be extracted, and so on down the chain of output units. Other algorithms have been suggested which attempt to find the principal *subspace*, i.e. the space spanned by the principal components, rather than the principal components themselves. This is still sufficient to minimize mean square reconstruction error and optimally preserve information [44], [65]: any linear rotation of the output components will not affect these properties. Therefore unless the principal components themselves are needed for a particular application, it may be sufficient to have some other set of vectors which spans the principal subspace.

If only the principal subspace is needed, this can be also achieved with the symmetrical version of the Williams [79] Symmetric Error Correction algorithm, which is equivalent to Oja's Subspace Network [55]. In this system, the re-estimator $\hat{\mathbf{x}}_j(t)$ in (15) or (16) is replaced with

$$\hat{\mathbf{x}}_j(t) = \hat{\mathbf{x}}(t) = \sum_{k=1}^{M} \mathbf{w}_k(t) y_k(t). \tag{17}$$

With the algorithm (14), (17), the network will converge to a set of outputs which span the principal subspace, rather than the principal components themselves, but with the weight vectors $\mathbf{w}_j$ orthonormal at convergence (as they are for the GHA and SGA algorithms).

Oja, Ogawa and Wangviwattana [58] modified this algorithm to add an additional symmetry-breaking factor to give

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \epsilon y_j(t)(\mathbf{x}(t) - \theta_j \hat{\mathbf{x}}(t)) \tag{18}$$

with $0 < \theta_1 < \theta_2 < \cdots < \theta_M$ and $\hat{\mathbf{x}}(t)$ given by (17). This favours lower-numbered units over higher-numbered units, resulting in the true principal components being extracted rather than the just the principal subspace.

Decorrelating algorithms An alternative approach to using the re-estimator $\hat{\mathbf{x}}_j(t)$ to remove the effect of other principal components, is to decorrelate successive outputs from earlier outputs. This forces an output to learn to respond to different principal component from other outputs. Rubner and Tavan [66], for example, suggest such a hierarchical decorrelating model with adaptive lateral inhibition from lower-numbered output units to higher-numbered output units (Fig. 2).

The outputs in this system are give by

$$y_j(t) = \mathbf{w}_j(t).\mathbf{x}(t) + \sum_{k=1}^{j-1} v_{jk}(t) y_k(t) \tag{19}$$

which means that the outputs y_j are calculated in the order $y_1, y_2, \ldots, y_M$. Each forward weight vector $\mathbf{w}_j$ is updated according to the Oja Rule (either (9)–(10) or (11)), and with the lateral connections v_{jk} to output y_j from y_k updated according to the simple anti-Hebbian algorithm

$$v_{jk}(t+1) = v_{jk}(t) - \epsilon_v y_j(t) y_k(t) \tag{20}$$

for $j > k$. This algorithm for the lateral connections is *anti*-Hebbian since the weight value *decreases* (or the inhibition increases) in proportion to the product of the activations of the units at either end.

Again, the output from the first unit y_1 finds the first principal component, since it is calculated according to the Oja Rule with no decorrelating terms. The inhibition between this unit and y_2 increases according to (20) until this second output is decorrelated from y_1, forcing y_2 to find the second principal component. The third output y_3 is decorrelated from the previous two, and so on until all M desired principal components have been extracted in order. Kung and Diamantaras [40] suggested an Adaptive Principal Component Extractor (APEX) which is related to the Rubner and Tavan approach.

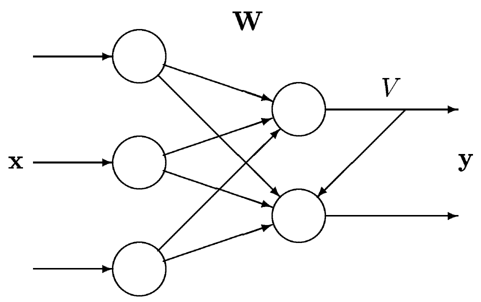

Fig. 2. PCA network with hierarchical decorrelation of output units.

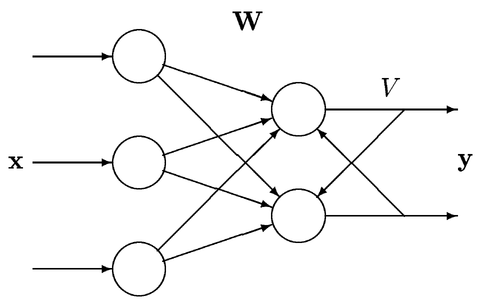

Fig. 3. Földiák network with symmetrical decorrelation of output units.

Földiák [20] suggested that units obeying the Oja rule should be decorrelated using a *symmetric* decorrelating stage (Fig. 3).

which in matrix notation is

$$\mathbf{y}(t) = \mathbf{W}(t)\mathbf{x}(t) + \mathbf{V}(t)\mathbf{y}(t) \tag{21}$$

where the lateral inhibition matrix $\mathbf{V}(t)$ is forced to have zero entries on the diagonal. If the outputs of the network are allowed to settle so that (21) is satisfied, then we get

$$\mathbf{y}(t) = (\mathbf{I} - \mathbf{V}(t))^{-1}\,\mathbf{W}(t)\mathbf{x}(t) \tag{22}$$

where $\mathbf{I}$ is the identity matrix, provided that all the eigenvalues of $\mathbf{V}(t)$ are less than unity [30]. In practice, it may be sufficient to use the first two terms of the expansion [42]

$$(\mathbf{I} - \mathbf{V}(t))^{-1} \approx \mathbf{I} + \mathbf{V}(t) + \cdots \tag{23}$$

in an artificial system. Once settled, the Oja rule (11) is used to update each forward weight vector, and the lateral connections v_{jk} are updated with the anti-Hebbian algorithm (20) as for the Rubner and Tavan model, but this time for all the symmetric connections $j \neq k$. In this network, the lateral connection matrix $\mathbf{V} = [v_{ij}]$ is not forced to be subdiagonal. Note that the update of v_{jk} is symmetrical, so if $\mathbf{V}$ is initially symmetrical, it will remain so for all t.

Leen [42] analyzed the stability of Födiák's network, and suggested that the algorithm (20) for the lateral connections be modified to add a term proportional to the lateral connection weight itself

$$v_{jk}(t+1) = v_{jk}(t) + \epsilon_w \beta v_{jk} - \epsilon_v y_j(t) y_k(t) \tag{24}$$

or an activity-dependent term

$$\begin{aligned} v_{jk}(t+1) \;=\; & v_{jk}(t) \qquad (25)\\ & +\epsilon_w \left(\left\langle y_j(t)^2 + y_k(t)^2 \right\rangle\right) v_{jk} \\ & -\epsilon_v y_j(t) y_k(t). \end{aligned}$$

where the $\langle y_j(t)^2 + y_k(t)^2 \rangle$ term means that the variance of the output units has to be accumulated as a measure of activity to drive this algorithm. With the latter modification, the weight vectors converge to the principal components themselves, rather than just the principal subspace, provided that $\epsilon_v > 2\epsilon_w$ [42]. Since there is nothing in this system that favours any one output over any other, the principal components found by this system will not appear in any particular order.

Orthonormalized principal subspace Almost all of the previous algorithms are generalizations of the Oja Rule (11), and consequently tend to preserve the variance of the input components at the output. However, this is not necessary from either the information theory or minimum reconstruction error viewpoints, so alternatives are possible.

For example, Plumbley [64] suggested two networks which can extract the principal subspace, but produce uncorrelated equal variance outputs, i.e. $\mathbf{Q}^{\mathbf{y}} \to \beta\mathbf{I}$.

Having outputs of equal variance is desirable in systems with limited channel capacity, and may facilitate subsequent classification by normalizing the scale of the outputs.

The first (Fig. 4(a)) uses lateral inhibition at the output, similar to the Földiák network (21), but with self-inhibitory connections v_{ii} allowed. As for that network, we have

$$\mathbf{y}(t) = \mathbf{W}(t)\mathbf{x}(t) + \mathbf{V}(t)\mathbf{y}(t) \tag{26}$$

leading to

$$\mathbf{y}(t) = (\mathbf{I} - \mathbf{V}(t))^{-1}\,\mathbf{W}(t)\mathbf{x}(t) \tag{27}$$

after settling, but with diagonal entries of $\mathbf{V}(t)$ allowed. Using the algorithm

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \epsilon_w \left(y_j(t)\mathbf{x}(t) - \alpha \mathbf{w}_j(t)\right) \tag{28}$$

and

$$v_{jk}(t+1) = v_{jk}(t) + \epsilon_v \left(y_j(t) y_k(t) - \beta \delta_{jk}\right) \tag{29}$$

where δ_{jk} is the Kronecker delta which is 1 when $j = k$ and 0 otherwise, and $\epsilon_v \gg \epsilon_w$, the outputs will converge to an uncorrelated, equal variance set which spans the principal subspace.

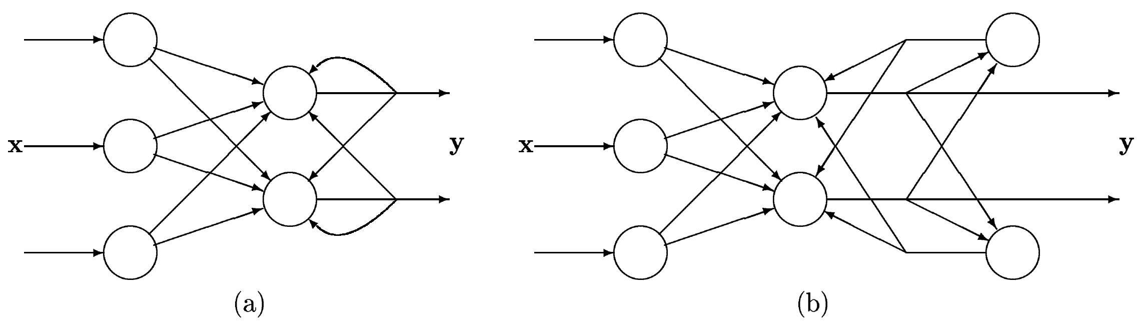

Fig. 4. PCA networks with uncorrelated equal variance outputs.

The second network (Fig. 4(b)) uses a set of inhibitory interneurons, and is specified by

$$\begin{aligned} \mathbf{y}(t) &= \mathbf{W}(t)\mathbf{x}(t) - \mathbf{V}(t)\mathbf{z}(t) \qquad (30) \\ \mathbf{z}(t) &= \mathbf{V}(t)^T\mathbf{y}(t) \end{aligned}$$

leading to

$$\mathbf{y}(t) = \left(\mathbf{I} + \mathbf{V}(t)\mathbf{V}(t)^T\right)^{-1}\mathbf{W}(t)\mathbf{x}(t) \qquad (31)$$

after settling. Using the algorithm (28) again for the forward weights, with the algorithm

$$\begin{aligned} v_{jk}(t+1) &= v_{jk}(t) + \qquad (32) \\ &\epsilon_v\left(y_j(t)z_k(t) - \beta v_k(t)\right) \end{aligned}$$

for the lateral connections, again the algorithm converges when the outputs are uncorrelated, equal variance, and span the principal subspace (provided that $\epsilon_v \gg \epsilon_w$, and all the selected principal components of the input have variance greater than α [64]).

Auto-Encoders Instead of using one of the many principal subspace algorithms mentioned so far, the Error Back Propagation ('BackProp') algorithm can also be used to find the principal subspace [4]. Consider the $N-M-N$ linear network of Fig. 5 with

$$\begin{aligned} y_j(t) &= \sum_{i=1}^{N} w_{ji}(t)x_i(t) \qquad (33) \\ z_k(t) &= \sum_{j=1}^{M} v_{kj}(t)y_j(t). \qquad (34) \end{aligned}$$

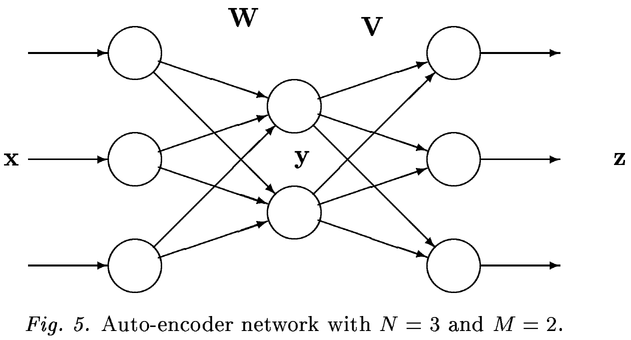

Fig. 5. Auto-encoder network with $N=3$ and $M=2$.

If BackProp [67] is used to minimize the mean squared error $E = \langle|\mathbf{x}(t) - \mathbf{z}(t)|^2\rangle$ in this system, the units in the hidden layer $\mathbf{y}$ will become some set of vectors which spans the principal subspace. Note that there is nothing to force the weight vectors $\mathbf{w}_i(t)$ to be orthogonal: any set of weights which spans the principal subspace is sufficient [4].

In common with other networks which use BackProp, the units in an auto-encoder can have nonlinear activation functions, such as the sigmoid function $\sigma(a) = 1/(1+\exp(a))$. In this case (33) and (34) become

$$\begin{aligned} y_j(t) &= \sigma\left(\sum_{i=1}^{N} w_{ji}(t)x_i(t)\right) \qquad (35) \\ z_k(t) &= \sigma\left(\sum_{j=1}^{M} v_{kj}(t)y_j(t)\right). \qquad (36) \end{aligned}$$

In theory these are capable of more complex behaviour than purely linear networks, and should be able to extract a nonlinear component or hypersurface from the input subspace. However, opinions are divided as to whether this is achieved in practice, and some have argued that a method based on SVD is more reliable, and not susceptible to local minima [12]. A further problem is that the nonlinear hidden layer codes are usually difficult to interpret. Saund [70] and Zemel and Hinton

[80] have explored ways of constraining the hidden unit codes adopted by autoencoder networks to obtain more biologically plausible and/or interpretable representations such as value codes or topographic maps.

2.2. Temporal prediction

The techniques described so far only deal with static patterns. An important characteristic of real-world data such as visual and speech signals is the presence of temporal structure. By discovering ways in which such structure is predictable over time, it is possible to form a compressed representation of the data, and thereby capture its underlying temporal constraints. One way to pose this problem in an unsupervised learning framework is to train a network to predict its input at the next time step based on previous inputs. This is a generalization of the auto-encoder network in which the error signal is $E = \langle |\mathbf{x}(t-1) - \mathbf{z}(t)|^2 \rangle$. The learned mapping is information-preserving to the extent that each input pattern can be accurately predicted from inputs at previous time steps. The two major techniques for providing temporally varying information to a network are to either use tapped delay lines - making inputs from several time steps available simultaneously, thereby spatializing the time domain - or using recurrent feedback connections; a number of variations on architectures and training methods for these networks are reviewed by Mozer [49]. Unlike delay-line neural networks, recurrent networks are sensitive to information over a potentially infinite time span; however, in practice they have enormous difficulty learning to maintain state information over long time intervals. Schmidhuber [72] and Mozer [48] have proposed two different ways of addressing this problem in recurrent networks.

2.3. Independent Components Analysis

When the input is composed of a combination of independent signals, linear methods such as principal components analysis, in general, are incapable of separating the independent sources. Jutten and Herault [34] developed a learning procedure, INCA, for extracting statistically independent components of the input vector, when the input vector $\mathbf{x}$ is modelled as an additive mixture of unknown independent signal components $\mathbf{x}(t) = \mathbf{A}\mathbf{z}(t)$ where the matrix $\mathbf{A}$ consists of unknown real scalars. A recursive network very much like Foldiak's symmetrical net (Fig. 3) is used to produce outputs $y_i(t) = x_i(t) - \sum_{k \neq i} v_{ik}(t) y_k(t)$ which come to approximate the independent signal components. Using the fact that if all higher order moments of two signals are uncorrelated, the signals are independent, a modified Hebb-rule is proposed:

$$\begin{aligned} v_{jk}(t) &= v_{jk}(t-1) \\ &\quad + \epsilon f(y_j(t)) g(y_k(t))) \end{aligned} \tag{37}$$

This rule decorrelates higher-order moments f and g of the inputs. One choice of functions used in many of Jutten and Herault's simulations, which they claim worked well for most of their test problems, is $f(x) = x^3$, $g(y) = \tan^{-1}(y)$.

2.4. Direct minimization of information loss

The methods discussed above ensure that minimal information is lost in a network mapping by minimizing reconstruction error. A related, but more general approach is to use concepts from information theory. A number of learning procedures have been proposed which minimize the information loss in a network mapping, subject to processing constraints. The common feature of these methods is the preservation of mutual information between the input vector $\mathbf{x}$ and output vector $\mathbf{y}$:

$$I_{x;y} = H(\mathbf{x}) + H(\mathbf{y}) - H(\mathbf{x}, \mathbf{y}) \tag{38}$$

where $H(\mathbf{x}, \mathbf{y})$ is the entropy of the joint distribution, $H(\mathbf{x}, \mathbf{y}) = -\int_{\mathbf{x},\mathbf{y}} p(\mathbf{x}, \mathbf{y}) \log p(\mathbf{x}, \mathbf{y})$. This measure is due to Shannon [74] and tells us the amount of information in $\mathbf{x}$ less the amount remaining in $\mathbf{x}$ when $\mathbf{y}$ is known, or the uncertainty in $\mathbf{x}$ which is accounted for by $\mathbf{y}$ (and vice versa). In the unconstrained noise-free case, all the information can be preserved simply by copying the input. Linsker [44] proposes maximizing the information rate in the presence of processing noise at either the input or output layer (the "Infomax principle"). The information rate for a collection of linear units with Gaussian input distribution and independent, equal-variance Gaussian noise added to the outputs is:

$$I = 0.5 \log \left(\frac{|\mathbf{Q}^{\mathbf{y}}|}{V(n)^N} \right) \tag{39}$$

where $|\mathbf{Q}^{\mathbf{y}}|$ is the determinant of the covariance matrix of the output vector $\mathbf{y}$ (the signal plus noise) and $V(n)$ is the noise variance. This results in a tradeoff between maximizing the variances of the outputs, and decorrelating them, depending on the noise level.

The above analyses apply to linear networks. Adding the usual form of sigmoid nonlinearity makes an information-theoretic analysis much more difficult. Linsker [45] shows that adding a "weakly nonlinear" (cubic) input-output relation to the Infomax principle, for certain translation-invariant input distributions, results in a set of units tuned to different spatial frequencies and spatial locations, much like a wavelet representation.

An alternative optimality criterion proposed by Barlow [5] is to find a minimally *redundant* encoding, which should facilitate subsequent learning. 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). Thus, such a representation should be useful as a preprocessing stage for a variety of problems.

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. In the general case, this problem is intractable. Atick and Redlich [3] have proposed a cost function for Barlow's principle for linear systems which minimizes the power (redundancy) in the outputs subject to a minimal information loss constraint. This is closely related to Plumbley's [63] objective function, which minimizes the information loss subject to a fixed power constraint, and for which a simple Hebbian learning scheme is derived. Schmidhuber [71] has proposed several ways of approximating Barlow's minimum redundancy principle in the general case, for nonlinear networks. Without the Gaussian assumptions, this implies a much stronger result of statistically independent, rather than just decorrelated, outputs. Schmidhuber's learning scheme is rather complex, however, and appears to be subject to oscillations and local minima.

2.5. Applications

Many of the applications for PCA and subspace methods are related to data compression or preprocessing of speech [18], [12] or images [15]. Sanger, for example, used his GHA network for image compression [69]. He also extended this to nonlinear units with a rectification nonlinearity, which he used to discover stereo disparity in random dot stereograms.

Leen, Rudnick and Hammerstrom [43] used PCA networks for signal pre-processing and found improved classification performance. Karhunen and Joutsensalo [36] use the SGA principal subspace algorithm for frequency estimation. They suggest that an undate factor of $\epsilon \approx 1/\left(0.5|\mathbf{x}(t)|^2\right)$ gives good initial convergence, with ϵ decreasing with $1/t$ after the first few cycles.

An alternative application of subspace methods is for more direct classification. Oja [55] allows a network to learn one subspace per class, with modifications to the learning algorithms to make sure the subspaces differentiate between classes. When classifying, the class with the closest subspace to the input vector (calculated from its projection into each subspace) is the 'winner'. He used this approach to classify Brodatz image textures.

Temporal sequence predictors have been applied to a range of prediction problems. Recurrent networks have been used predominantly for symbolic tasks in which the goal is to represent discrete hidden variables by extracting temporal structure. For example, Elman [17] has used simple recurrent networks to infer grammatical structure from sentences. Mozer [49] has applied recurrent networks with multiple time decay constants to the problem of extracting musical structure at multiple time scales by predicting notes in a musical score.

The tapped-delay-line method has been used successfully on real-valued, noisy signal prediction problems such as chaotic time series prediction [41], [78]. The key to selecting the appropriate architecture for a temporal problem is to have an appropriate characterization of the problem in advance. For example, if the problem requires explicit knowledge of previous real-valued inputs within a short time window, the time-delay architecture is appropriate. If the problem requires only a few bits of history information over arbitrarily long time periods, a small number of hidden units with recurrent connections can be used to learn these "hidden state variables". Accurate real-valued state information is difficult to maintain over time in recurrent networks unless a very large number of hidden units are used.

Jutten and Herault [34] applied their independent components analysis algorithm to images of sloped handwriting. When random samples of (x, y) points were used as input, a two-neuron network was able to remove the dependence between the input coordinates, resulting in output images consisting of unsloped text. Further, in a companion paper, Jutten and Herault [35] demonstrated the success of the algorithm on a number of difficult synthetic nonlinear source separation problems.

3. Density Estimation Techniques

Rather than trying to retain all of the information contained in the input, we could try to develop a more abstract representation by characterizing its underlying probability distribution. This approach can lead to useful features for classification or other subsequent learning tasks. Many standard statistical methods fall under the category of density estimation techniques, and several unsupervised learning procedures can be viewed in this way. (In fact, the techniques discussed here under the category of density estimation might more specifically be characterized as maximum likelihood parameter estimation methods.) The general approach is to assume a prior model which constrains the general form of the probability density function, and then to search for the particular model parameters defining the density function most likely to have generated the observed data. This approach can be mapped onto an unsupervised learning problem if we treat the network weights as the model parameters, and the overall function computed by the network as being directly related to the density function.

3.1. "1-of-n encodings": Mixture models and competitive learning

One possible choice of prior model is a mixture of Gaussians. The underlying assumption in this case is that each data point was actually generated by one of n Gaussians having different means μ_i, variances ${\sigma_i}^2$, and prior probabilities or mixing proportions π_i. Fixing the model parameters μ_i, σ_i, and π_i, we can compute the probability of a given data point $\mathbf{x}$ as follows:

$$\begin{aligned} & p(\mathbf{x}|\left\{\mu_i\right\}, \left\{\sigma_i\right\}, \left\{\pi_i\right\}) \\ & \quad = \textstyle\sum_{i=1}^n \pi_i P_i(\mathbf{x}, \mu_i, \sigma_i) \end{aligned} \tag{40}$$

where $P_i(\mathbf{x}, \mu_i, \sigma_i)$ is the probability of $\mathbf{x}$ under the ith Gaussian. Applying Bayes' rule, we can also compute the probability that any one of the Gaussians generated the data point $\mathbf{x}$:

$$\begin{aligned} & p(i|\mathbf{x}, \left\{\mu_i\right\}, \left\{\sigma_i\right\}, \left\{\pi_i\right\}) \\ & \quad = \frac{\pi_i P_i(\mathbf{x}, \mu_i, \sigma_i)}{\sum_{j=1}^n \pi_j P_j(\mathbf{x}, \mu_j, \sigma_j)} \end{aligned} \tag{41}$$

Given these probabilities, the model parameters can be adapted by performing gradient ascent in the log likelihood of the data given the model, $\log(L) = \sum_{\mathbf{x}} \log(p(\mathbf{x} \mid \left\{\mu_i\right\}, \left\{\sigma_i\right\}, \left\{\pi_i\right\}))$ The EM algorithm [16] alternately applies equation (41) (the Expectation step) and adapts the model parameters (the Maximization step) to converge on the maximum likelihood mixture model of the data.

Competitive learning procedures [76], [22], [37], [14], [68] are used primarily for clustering. The general idea underlying competitive learning is to induce a competition between units' responses, either by a winner-take-all activation function or with lateral interactions, so that only one unit in each competitive cluster tends to be active at a time. Typically only the winning unit learns on each case, by moving its weight vector closer to the current input pattern. For example, Rumelhart and Zipser's [68] version of competitive learning sets the activity of the winning unit (the one with the greatest total input, x_i) to one, and the rest to zero, and uses the following learning rule:

$$\begin{aligned} w_{ji}(t+1) &= w_{ji}(t) \qquad (42) \\ &+ \begin{cases} \varepsilon\left(\frac{x_i^\alpha}{\sum_k x_k^\alpha} - w_{ji}(t)\right) \\ \quad \text{if unit } j \text{ wins} \\ \quad \text{on pattern } \alpha \\ 0 \;\; \text{otherwise} \end{cases} \end{aligned}$$

By redistributing some proportion ε of the 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. Each unit is performing gradient descent in the squared distance between its weight vector and the patterns nearest to its weight vector (subject to a unit length constraint), which in the batch version is equivalent to the standard k-means clustering algorithm. Nowlan [53] has pointed out that this standard version of competitive learning is closely related to fitting a mixture of Gaussians model, with equal priors π_i and equal variances σ_i^2. Using the EM algorithm, every unit (not just the winner) would adjust its mean according to its distance to the current input vector, but in proportion to the probability that its Gaussian model accounts for the current input (equation (41)). Competitive learning approximates this step by making a hard (one-in-n-ary) decision as to which unit accounts for the input. Thus, the same learning rule applies, except that the proportional weighting is replaced by an all-or-none decision. So hard competitive learning can be viewed as a mixture of Gaussians estimation procedure using infinitesimally small variances.

Nowlan [52], [53] proposed a "soft competitive learning" model for neural networks. 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, $P_i(\mathbf{x}(t), \mu_i, \sigma_i)$:

$$\begin{aligned} &\mu_i(t) = \qquad (43) \\ &\frac{\kappa_i(t-1)\mu_i(t-1) + P_i(\mathbf{x}(t), \mu_i, \sigma_i)\mathbf{x}(t)}{\kappa_i(t-1) + P_i(\mathbf{x}(t), \mu_i, \sigma_i)} \end{aligned}$$

The variances can be adapted similarly. This is an online version of the EM algorithm for Gaussian densities with equal priors, and adaptive means and variances. Neal and Hinton [51] have shown that incremental variants of EM perform gradient descent in a global energy function; they treat Nowlan's algorithm as a special case of these, and show that it performs approximate gradient descent.

An appealing aspect of the soft competitive learning algorithm is that it can be generalized to incorporate arbitrary prior information such as supervisory signals, as in the "competing experts" model of Jacobs et al. [32]. Jordan and Jacobs [33] show how to generalize the competing experts framework to discover a hierarchical decomposition of structure in the input.

One useful variation on competitive learning is to define a neighborhood relation among the units, for example, by arranging them on a two-dimensional lattice; each unit then learns in proportion to its distance from the winning unit [76], [37]. This forces the competing units to form a *topographic* mapping in which nearby points in the input space are mapped to nearby points in the neighborhood space. In Kohonen's model of unsupervised topological map formation [37], [38] a set $N_i(t)$ is defined for each unit i, determining those units within its neighborhood. 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:

$$w_{ji}(t+1) = w_{ji}(t) \begin{cases} \varepsilon(x_i^\alpha - w_{ji}(t)) \\ \text{if unit } j \in N_c, \\ 0 \;\; \text{otherwise} \end{cases} \qquad (44)$$

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. The neighborhood set N_c can be replaced by a continuous function $N(i,j)$ of the distance between units i and j in the lattice; this leads to smoother learning. For neighborhoods of size 1, like standard competitive learning, Kohonen's algorithm is equivalent to k-means clustering [38]. 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. Luttrell [46] has shown how Kohonen's algorithm can be viewed as a variant of minimum distortion vector quantization (MDVQ). If we interpret the weight vector of the winning unit as the predicted input, $\hat{\mathbf{x}} = \mathbf{w}_c$, then the distortion or reconstruction error is just

$$D = |\mathbf{x} - \hat{\mathbf{x}}| \tag{45}$$

Now suppose there is an additive noise process π which distorts the output of units, making it impossible to determine with complete certainty which output was the winner. Then each unit will contribute to the distortion error, as a function of its distance to the winner:

$$D = \sum_i \pi(y_i - y_c)\, |\mathbf{x} - \hat{\mathbf{x}}| \tag{46}$$

This noisy version of MDVQ is equivalent to Kohonen's algorithm, where π is the neighborhood function; within the same framework, Luttrell [46] goes on to generalize the Kohonen algorithm to perform hierarchical MDVQ. In summary, like the hard competitive learning model, Kohonen's algorithm can be viewed as a version of the mixture of Gaussians model with infinitesimally small variances, with the addition of a noise process distorting our estimates of the probabilities of each Gaussian capturing the data.

Fukushima's Neocognitron [23] generalizes standard (hard) competitive learning to form a multi-resolution hierarchy of translation-invariant feature detectors. This network has produced impressive results on translation- and scale-invariant character recognition. The architecture of the network is as follows: each layer is organized along two dimensions, 1) into "planes" of units having different receptive fields but weights constrained to be identical, and 2) into "columns" of units having overlapping receptive fields but different weights. Units in higher layers receive input from a spatially localized region of units (in all planes) in the layer below, hence a hierarchy of receptive fields of different spatial scales is formed. The details of the unit activation functions are rather complex; the essential features of the model are that 1) units within a column compete to respond via lateral inhibition, and 2) units within a spatially localized region within a plane transmit the "inclusive or" of their responses to the next layer up. Hence, if the feature encoded by a particular plane is detected anywhere in the input "retina" it gets transmitted to the next layer up but with some loss of spatial localization. As in competitive learning, the winning unit within a competing cluster (column) moves its weights toward its current input; additionally, the other units in the plane perform identical weight updates. Note that it is possible for multiple planes to adapt simultaneously on a given case. Thus, the Neocognitron is another variation on density estimation, that is, it is equivalent to hard competitive learning with the addition of equality constraints between weights of corresponding units in different competing clusters having identical receptive fields.

3.2. Combinatorial representations

A major limitation of mixture models and standard competitive learning schemes is that they employ a 1-of-n encoding, in which a single unit or model is assumed to have generated the data. A *multiple causes* model is more appropriate when the most compact data description consists of several independent parameters (e.g. color, shape, and size of an object in a visual scene).

Mozer [47] has proposed a distributed version of competitive learning for binary units that discovers multiple classifications of the data. Each layer of competing units receives as input the reconstruction error from the previous layer, and performs clustering on that error. Mozer shows that this scheme can be used to discover useful distributed binary features for image compression.

A more general way to learn arbitrary probability distributions over data is with the stochastic Boltzmann Machine (SBM) [29], or its computationally more efficient cousin the Deterministic Boltzmann Machine (DBM) [61]. The DBM is considerably faster to train, as the mean field approximation eliminates the stochasticity in both the annealing process, and the sampling of the equilibrium correlation statistics. Theoretically one could train an unsupervised DBM exactly as for the SBM. Unfortunately, the mean field approximation used in DBMs is not particularly useful in this case; a DBM cannot form an adequate representation of the unclamped distribution, as all 2^n states of the n units are represented by a single mean state [25]. Peterson and Hartman [62] suggest one way around this problem. In the negative phase, only a random subset of the units are left unclamped. Using this method, they showed that the network was able to learn a set of random patterns, and perform pattern completion when partially specified or noisy patterns were presented.

Freund and Haussler [21] describe an efficient way to train SBMs unsupervised. The goal is to learn the "hidden causes" of a collection of patterns, with each hidden unit representing one hidden cause. They use a restricted architecture, allowing only between-layer connections, so that when all of the visible units are clamped, no settling is required. For this case, 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. 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 [50] presents a more general way of modeling the probability distribution of a pattern set in a multilayer stochastic "connectionist belief network", which falls within the class of Pearl's belief networks [59]. As it is not restricted to single layer networks, it should be able to model complex distributions with fewer parameters, although with a high price in learning time.

3.3. Applications

Variations on soft competitive learning, also referred to as radial basis function (RBF) networks, are now being widely used to preprocess speech data. A common technique is to use unsupervised learning to adapt the Gaussian means (the mixing proportions are subsequently thrown out), and then add a second layer of linear units which are trained via supervised learning to discover optimal linear combinations of the Gaussian unit outputs to perform classification. For example, Nowlan [52] showed that this method is superior to the traditional "hard competitive learning models" on two classification tasks, hand-written digit and vowel recognition. Jordan and Jacobs [33] applied the hierarchical version of the competing experts model in a set of competing auto-encoder networks that learn a hierarchical classification of leaf morphology data.

Kohonen [38] has applied his 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. In applications where clustering is an appropriate preprocessing stage, Kohonen's algorithm is easy to implement and tends to be less sensitive to initial conditions compared to standard competitive learning; the latter is prone to greedy behaviour, where a subset of the competing units capture all of the patterns initially, and the other competitors are thus prevented from ever learning.

The Neocognitron has been applied to simple character recognition problems [24]. Because of the equality constraints implicit in the learning procedure, the model is able to learn classes which are somewhat scale- and shift-invariant. Thus, the algorithm has proven to be able to discriminate nonlinearly separable pattern classes by building in translation-invariance of feature detectors.

The methods proposed by Freund and Haussler [21] and Neal [50] are both of theoretical interest, and could have potential applications in unsupervised higher-order feature discovery for classification. Their efficient use of such methods in real-world applications, however, remains to be demonstrated.

4. Feature extraction methods

The methods discussed in the previous two sections are based on the goal of characterizing as accurately as possible the input patterns, or the underlying distribution which generated them. This is a reasonable first step in extracting useful structure from data, assuming minimal prior knowledge. Often real-world data is redundant and noisy, so general methods like clustering or principal components are useful for improving the signal to noise ratio and achieving data compression. But how can unsupervised learning be applied beyond these preprocessing stages, to extract higher order features and build more abstract representations? One approach is to build in more sophisticated prior models; we have already seen several examples of this approach (e.g. Freund and Haussler's [21] and Neal's [50] methods). Another approach is to restrict our search to particular kinds of structure. If we can make constraining assumptions about the kind of structure we are looking for, we can build these constraints into the network's architecture and/or objective function and thereby develop more efficient, highly specialized learning procedures. In this last section, we consider several examples of learning procedures based on the idea of learning particular features of the data.

4.1. Maximally "selective" projections

Bienenstock, Cooper and Munro [11] proposed a learning rule (commonly referred to as the BCM rule) which results in a form of temporal selectivity of a single unit 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:

$$\mathrm{Sel}(y_j) = 1 - \frac{\overline{y_j}}{\max(y_j)} \tag{47}$$

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 satisfy:

$$\dot{w}_{ji} = \Phi(y_j, \overline{y_j}) y_i - \epsilon w_{ji} \tag{48}$$

where $\dot{w}$ denotes the rate of change of w over time, $\overline{y_j}$ is the mean output, and Φ must satisfy:

$$\begin{aligned} &\mathrm{sign}\left(\Phi(y_j, \overline{y_j})\right) \\ &\quad = \mathrm{sign}\left(y_j - \left(\frac{\overline{y_j}}{c_o}\right)^p \overline{y_j}\right) \\ &\quad \text{if } y_j > 0 \end{aligned} \tag{49}$$

and

$$\Phi(0, \overline{y_j}) = 0 \quad \text{for all } \overline{y_j} \tag{50}$$

where p and c_o are positive constants, and units' activities are always positive. This learning rule led to the development of tight orientation tuning curves for units, using simple oriented line patterns as inputs. Intrator [31] has proposed a related objective function for maximizing selectivity, which causes a unit to discover projections of the data having bimodal or multi-modal distributions. Intrator's objective function for units with a sigmoidal nonlinearity $\sigma(\mathbf{x})$ is:

$$\begin{aligned} -\frac{\mu}{3} &\left\{\sigma^3\left(\mathbf{x.m}\right)\right. \\ &\left.-E\left[\sigma^2(\mathbf{x.m})\right]\sigma^2(\mathbf{x.m})\right\} \end{aligned} \tag{51}$$

where the threshold of a unit is set to $E\left[\sigma^2(\mathbf{x.m})\right]$. When applied to a group of units which inhibit each other, this leads to the discovery of multiple features in the data. Intrator discusses the relation of this method to exploratory projection pursuit. The method tends to discover projections having a non-Gaussian, or skewed distribution which may be useful as features for certain classification problems.

4.2. Statistical dependencies between the inputs

The GMAX algorithm [60] is based on the goal of redundancy-detection. It 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 asymmetric divergence between these two distributions is maximized:

$$G = P \log \frac{P}{Q} + (1-P) \log \frac{1-P}{1-Q} \tag{52}$$

In general, it is not feasible to calculate Q explicitly, as it requires sampling all 2^n possible states of the n input units. Pearlmutter and Hinton approximate the expected value of the output y under the distribution Q by simulating a "negative learning phase" (somewhat analogous to that of the Boltzmann machine) in which input patterns with independent components are generated. When a unit is trained in this manner 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 approximations: 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. Although the GMAX principle may have limited applicability to general problems in the binary case (because of the sampling problem mentioned above), it is possible to apply the same principle in the continuous (Gaussian) case, as described in the next subsection; this results in an interesting algorithm which should be applicable to multi-layer nonlinear networks.

4.3. Invariant features

It has been suggested that the computation of invariant features about the world plays a fundamental role in human pattern recognition. This suggests that a reasonable goal of unsupervised learning is to develop invariance detectors: units that discover features of the input distribution which exhibit some form of invariance (e.g., a unit that finds a nontrivial linear combination of its inputs which is always zero). One attractive aspect of this view is that the actual output of an invariance detector would represent the extent to which the current input violates the network's model of the regularities in the world. This is an efficient way of transmitting information about the current input.

Several algorithms for learning invariant features of the input have been proposed. Kohonen and Oja [39] proposed a learning algorithm for a single unit which acts as a "novelty detector", by responding best to patterns which are orthogonal to the principal subspace of the input distribution. Fallside [19] proposed a learning procedure which implements a linear prediction filter: a unit receives inputs representing the values of a signal at several time frames, and tries to make its output zero by computing the sum of the signal at the current time slice and a linear combination of the signal values at previous time slices. Atick and Redlich [2] proposed an equivalent learning procedure for a spatial predicting unit, to model the development of retinal ganglion cell kernels. The latter two methods could both be applied in the nonlinear case as well; these methods are closely related to Imax, described in the next subsection.

Becker [6] has shown that a continuous generalization of GMAX for Gaussian input distributions results in an invariance detector that minimizes the ratio of its output variance divided by the variance that would be expected if the input lines were independent (the sum of the variances of the inputs):

$$\frac{V(y_i)}{\sum_j w_{ij} V(y_j)} \tag{53}$$

where $y_i = \sum_j w_{ij} y_j$. Although this analysis assumes that the output y_i is a linear function of the inputs y_j, the inputs could themselves be the outputs of nonlinear hidden units, resulting in a multi-layer learning procedure for discovering higher-order invariants.

This algorithm is further generalized to apply to a group of units which form a mixture model of different invariant properties of the input patterns [6]. Schraudolph and Sejnowski [73] propose a closely related learning scheme, combining a variance-minimizing anti-Hebbian term and a term that prevents the weights all converging to zero. They show that a set of competing units can thereby discover population codes for stereo disparity in random dot stereograms. Bell [10] has proposed an energy-minimizing algorithm for a single axon that leads to an anti-Hebbian learning rule that predicts the evolution of ion channel densities; this allows an axon to redistribute channels so as to efficiently process recurring spatio-temporal patterns.

4.4. Spatially coherent features

Becker and Hinton [8] proposed that a good objective function for unsupervised learning is to discover properties of the sensory input that exhibit coherence across space and time. The Imax learning procedure [8] does this by maximizing the mutual information between the outputs, y_a and y_b, of network modules that receive input from different parts of the sensory input (e.g. different modalities, or different spatial or temporal samples), as shown in Figure 6. To compute the entropies in $I_{y_a;y_b} = H(y_a)+H(y_b)-H(y_a,y_b)$, constraining assumptions about the distributions of y_a and y_b must be made.

In the discrete binary case, the expected outputs of two units can be approximated, as can their joint probabilities, by sampling over the input ensemble. The entropies can then be computed analytically:

$$\begin{aligned} H(y_i) &= -\langle \log p_i \rangle \\ &= -p_i \log p_i - p_{\bar{i}} \log p_{\bar{i}} \end{aligned} \tag{54}$$

where $p_i = <y_i>$ 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), and $p_{\bar{i}} = <(1-y_i)>$. The following learning rule is then obtained by differentiating

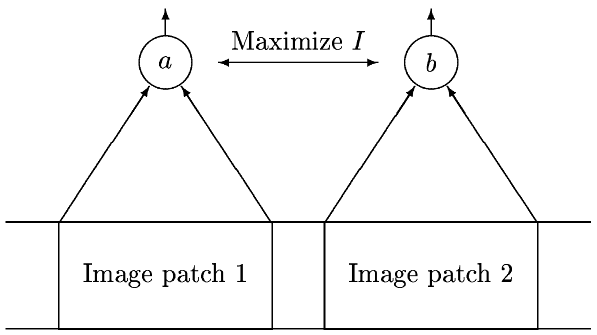

Fig. 6. Two units with separate inputs, that maximize their mutual information.

$I_{y_i;y_j}$:

$$\begin{aligned} \Delta w &= \sum_\alpha \varepsilon \frac{\partial I_{y_i;y_j}}{\partial p_i^\alpha} \frac{\partial p_i^\alpha}{\partial w} \\ &= -P^\alpha \left[\log \frac{p_i}{p_{\bar{i}}} - p_j^\alpha \log \frac{p_{ij}}{p_{\bar{i}j}} \right. \\ &\quad \left. - p_{\bar{j}}^\alpha \log \frac{p_{i\bar{j}}}{p_{\bar{i}\bar{j}}} \right] \frac{\partial p_i^\alpha}{\partial w} \end{aligned} \tag{55}$$

where p_i^α is the expected output of the ith unit on training case α, P^α is the probability of training case α, and $p_{ij} = <y_i y_j>$, etc. This learning rule can be applied to two multi-layer modules to learn features that are not linearly separable, such as the shift in random binary shift patterns [6]. The two modules receive as input random binary patterns in which the left half is a shifted version of the right half. The inputs to the two modules are unrelated apart from having the same shift. Unfortunately, at least on this particular problem, the binary version of Imax has a tendency to become trapped in local maxima, because the objective function encourages units to become very strongly binary (and hence, to develop very large weights).

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 [13]:

$$P(A = a_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}} \tag{56}$$

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:

$$\begin{aligned} I_{A;B} &= -\sum_i P(A = a_i) \log P(A = a_i) \\ &\quad -\sum_j P(B = b_j) \log P(B = b_j) \end{aligned}$$

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

This is a straightforward generalization of the binary case, and similar learning rules can be derived [6].

Under Gaussian assumptions, a simpler objective function for Imax learning can be derived. If we assume that both modules receive input that is caused by some common underlying signal s corrupted by independent Gaussian noise in each input patch, and that the modules transform the input into outputs y_a and y_b that are each noisy versions of the signal, $y_a = s + n_a$, $y_b = s + n_b$, then:

$$I = 0.5 \log \frac{V(y_a + y_b)}{V(y_a - y_b)} \tag{57}$$

where V stands for variance. This measure tells how much information the average of y_a and y_b conveys about the common underlying signal, i.e. the feature which is coherent in the two input samples.

Multi-dimensional features The continuous version of Imax can be generalized to extract multiple parameters, $\mathbf{y_a}$ and $\mathbf{y_b}$, that each represent multi-dimensional Gaussian signals with independent additive Gaussian noise. In this case, the following objective function can be maximized:

$$I_{\mathbf{y_a}+\mathbf{y_b};signal} = 0.5 \log \frac{|\mathbf{Q}_{\mathbf{y_a}+\mathbf{y_b}}|}{|\mathbf{Q}_{\mathbf{y_a}-\mathbf{y_b}}|} \tag{58}$$

where $\mathbf{y_a}$ and $\mathbf{y_b}$ are parameter vectors extracted from neighboring patches, $\mathbf{Q}$ is a covariance matrix and $|\mathbf{Q}|$ its determinant. Zemel and Hinton [81] have explored applications of this method to visual object recognition problems.

4.5. Applications

Intrator has demonstrated his nonlinear multi-unit version of the BCM rule on speech data having dimensionality as large as 5500; compared to supervised BackProp, the algorithm " ... is able to find a richer, linguistically meaningful structure, containing burst locations and formant tracking of the three different stops that allowed a better generalization to other speakers and to voiced stops." (Intrator [31], pp. 105).

The multi-valued discrete version of Imax has been successfully applied in single-layer networks to toy problems such as extracting combinations of spatial frequency and phase in sinusoidal intensity patterns [6]. It has also been applied, with limited success, by Lapedes and Steeg to the much more difficult problem of extracting mutually predictable features from protein sequences and their three-dimensional descriptions (Evan Steeg, 1993, personal communication); here, however, the method seemed to be prone to getting trapped in local optima.

The continuous version of Imax has mainly been applied to toy problems of sufficient complexity that they could not be solved by a single layer or linear network. For example, on the binary shift problem, it performs extremely well, and getting stuck in local minima does not seem to be a problem. It has also been applied successfully to continuous higher-order feature extraction problems such as learning to represent stereo disparity and surface curvature in random dot stereograms [8]. Under a mixture model of the underlying coherent feature, the algorithm can be extended to develop population codes of spatially coherent features such as stereo disparity [8], and to model the locations of discontinuities in depth [9]. Finally, it has been applied to temporally varying patterns to classify temporally coherent objects over time [7].

Zemel and Hinton [81] have applied the multi-dimensional version of Imax to the problem of learning to represent the viewing parameters of simple synthetic two-dimensional objects. The learning procedure tries to extract multiple features from an image patch which are *uncorrelated* with each other, as well as being good predictors of the feature vector extracted from a neighboring patch. The method is potentially more powerful than linear methods such as principal components analysis, because the network can compute arbitrary nonlinear transformations in order to extract these features. One difficulty with the method is the practical limitations of computing determinants of ill-conditioned matrices. Another drawback is that the representation learned by the algorithm, at least on the "viewing parameters" problem, is not all that simple. The features the network learns to represent are typically each nonlinear combinations of the viewing parameters (e.g., scale, location, and size), which cannot easily be interpreted by themselves.

Source Separation One promising real-world application for Imax and related methods is in signal separation, or contrast enhancement of two sources. This problem is complementary to coherence detection: rather than discovering the common underlying signal, we want to factor out the *separate* components in the two sources. One way to do this is to try to extract features having *minimal* mutual information; however, we must rule out trivial solutions in which the features convey no information at all. Ukrainec and Haykin [75] have proposed minimizing equation (57), adding a penalty term $\rho\,|\mathbf{Q}^{\mathbf{y}}| - 1$ to prevent degenerate solutions, where $\mathbf{y} = [y_a, y_b]$. Ukrainek and Haykin have applied this idea successfully to the problem of detecting distinctive features in a pair of orthogonally polarized radar detectors; the goal was to detect a reflector target in the dual polarized radar images. By learning the appropriate nonlinear mapping, this method was able to outperform standard linear preprocessing using PCA. Note that an alternative algorithm for source separation by Jutten and Herault [34] was discussed in section 2.3.

5. Conclusions

Unsupervised learning procedures have been applied to many real-world problems to reduce noise, compress data, and extract useful features for subsequent classification. We have attempted to provide a survey of the most widely used and successful of these methods, and to highlight methods that show promise for future work. The major challenge for future research in this area is to develop more powerful learning procedures which can extract nonlinear features that are applicable to a variety of signal classification problems.

Acknowledgements

The authors wish to thank Chris Williams, Rich Zemel, Peter Dayan and the anonymous reviewers for helpful comments on earlier drafts of this paper.

References

1. H. M. Abbas and M. M. Fahmy. A neural model for adaptive Karhunen Loéve transform (KLT). In *Proceedings of the International Joint Conference on Neural Networks, IJCNN-92 Baltimore*, pages II:975–980, 1992.
2. J. J. Atick and A. N. Redlich. Predicting ganglion and simple cell receptive field organizations from information theory. Technical Report IASSNS-HEP-89/55, Institute for Advanced Study, Princeton, 1989.
3. J. J. Atick and A. N. Redlich. Towards a theory of early visual processing. *Neural Computation*, 2:308-320, 1990.
4. P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1989.
5. H. B. Barlow. Unsupervised learning. *Neural Computation*, 1:295–311, 1989.
6. S. Becker. *An Information-theoretic Unsupervised Learning Algorithm for Neural Networks.* PhD thesis, University of Toronto, 1992.
7. S. Becker. Learning to categorize objects using temporal coherence. In *Advances in Neural Information Processing Systems 5*, pages 361–368. Morgan Kaufmann, 1993.
8. S. Becker and G. E. Hinton. A self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355:161–163, 1992.
9. S. Becker and G. E. Hinton. Learning mixture models of spatial coherence. *Neural Computation*, 5(2):267–277, 1993.
10. A. J. Bell. Self-organisation in real neurons: Anti-hebb in 'channel space'? In *Advances in Neural Information Processing Systems 4*, pages 59–66. Morgan Kaufmann, 1992.
11. E. L. Bienenstock, L. N. Cooper, and P. W. Munro. Theory for the development of neuron selectivity; orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2:32–48, 1982.
12. H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.
13. J.S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fougelman-Soulie and J. Herault, editors, *NATO ASI series on systems and computer science*. Springer-Verlag, 1990.
14. G.A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision*, 37:54–115, 1983.

15. G. W. Cottrell, P. W. Munro, and D. Zipser. Image compression by back propagation: A demonstration of extensional programming. In N. E. Sharkey, editor, *Advances in Cognitive Science*, volume 2. Abbex, Norwood, NJ, 1989.
16. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*, B-39:1–38, 1977.
17. J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.
18. J. L. Elman and D. Zipser. Learning the hidden structure of speech. ICS Report 8701, Institute of Cognitive Science, University of California, San Diego, 1987.
19. F. Fallside. On the analysis of multi-dimensional linear predicitve/autoregressive data by a class of single layer connectionist models. In *IEE Conference on Artificial Neural Networks*, pages 176–180, 1989.
20. P. Földiák. Adaptive network for optimal linear feature extraction. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN-89*, pages 401–405, Washington, DC, 1989.
21. Y. Freund and D. Haussler. Unsupervised learning of distributions on binary vectors using 2-layer networks. In *Advances In Neural Information Processing Systems 4*, pages 912–919. Morgan Kaufmann Publishers, 1992.
22. K. Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20:121–136, 1975.
23. K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.
24. K. Fukushima. A hierarchical neural network model for associative memory. *Biological Cybernetics*, 50:105–113, 1984.
25. C. Galland. *Learning in Deterministic Boltzmann Machine Networks.* PhD thesis, University of Toronto, 1992.
26. J. J. Gerbrands. On the relationships between SVD,KLT and PCA. *Pattern Recogntion*, 14:375–381, 1981.
27. G. H. Golub and C. F. Van Loan. *Matrix Computations.* North Oxford Academic, Oxford, 1983.
28. R. C. Gonzalez and P. Wintz. *Digital Image Processing.* Addison-Wesley, Reading, MA, second edition, 1987.
29. G. E. Hinton and T. J. Sejnowski. Learning and relearning in Boltzmann machines. In D. E. Rumelhart, J. L. McClelland, and the PDP research group, editors, *Parallel distributed processing: Explorations in the microstructure of cognition*, volume I, pages 282–317. Cambridge, MA: MIT Press, 1986.
30. K. Hornik and C.-M. Kuan. Convergence analysis of local feature extraction algorithms. *Neural Networks*, 5:229–240, 1992.
31. N. Intrator. Feature extraction using an unsupervised neural network. *Neural Computation*, 4(1):98–107, 1992.
32. R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1), 1991.
33. M. I. Jordan and R. A. Jacobs. Hierarchies of adaptive experts. In *Advances in Neural Information Processing Systems 5*, pages 985–992. Morgan Kaufmann, 1993.
34. C. Jutten and J. Herault. Blind separation of sources, part I: An adaptive algorithm based on enuromimetic architecture. *Signal Processing*, 24:1–10, 1991.
35. C. Jutten and J. Herault. Blind separation of sources, part II: Problems statement. *Signal Processing*, 24:11–20, 1991.
36. J. Karhunen and J. Joutsensalo. Tracking of sinusoidal frequencies by neural network learning algorithms. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP-91*, Toronto, Canada, 1991.
37. T. Kohonen. Clustering, taxonomy, and topological maps of patterns. In M. Lang, editor, *Proceedings of the Sixth International Conference on Pattern Recognition*, Silver Spring, MD, 1982. IEEE Computer Society Press.
38. T. Kohonen. The 'neural' phonetic typewriter. *IEEE Computer*, 21:11–22, 1988.
39. T. Kohonen and E. Oja. Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks of neuron-like elements. *Biological Cybernetics*, 21:85–95, 1976.
40. S. Y. Kung and K. I. Diamantaras. A neural network learning algorithm for adaptive principal component extraction (APEX). In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP-90*, pages II: 861–864, 1990.
41. A. S. Lapedes and R. M. Farber. Nonlinear signal processing using neural networks: Prediction and system modelling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, 1987.
42. T. K. Leen. Dynamics of learning in linear feature-discovery networks. *Network*, 2:85–105, 1991.
43. T. K. Leen, M. Rudnick, and D. Hammerstrom. Hebbian feature discovery improves classifier efficiency. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN-89*, pages I: 51–56, Washington, DC, 1989.
44. R. Linsker. Self-organization in a perceptual network. *IEEE Computer*, 21(3):105–117, March 1988.
45. R. Linsker. Deriving receptive fields using an optimal encoding criterion. In *Advances in Neural Information Processing Systems 5*, pages 953–960. Morgan Kaufmann, 1993.
46. S.P. Luttrell. Hierarchical vector quantisation. In *Proceedings of the Inst. of Elec. Eng.*, volume 136, pages 405–413, 1989.
47. M. C. Mozer. Discovering discrete distributed representations with iterative competitive learning. In *Advances in Neural Information Processing Systems 3*, pages 627–634. Morgan Kaufmann, 1991.
48. M. C. Mozer. Induction of multicale temporal structure. In *Advances in Neural Information Processing Systems 4*, pages 275–282. Morgan Kaufmann, 1992.
49. M. C. Mozer. Neural net architectures for temporal sequence procesing. In A. Weigend and N. Gershenfeld, editors, *Predicting the future and undertanding the past.* Redwood City, CA: Addison-Wesley Publishing, 1993.

50. R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.
51. R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. Submitted for publication.
52. S. J. Nowlan. Maximum likelihood competitive learning. In D. S. Touretzky, editor, *Neural Information Processing Systems, Vol. 2*, pages 574–582, San Mateo, CA, 1990. Morgan Kaufmann.
53. S. J. Nowlan. *Soft Competitive Adaptation: Neural Network Learning Algorithms based on Fitting Statistical Mixtures.* PhD thesis, Carnegie-Mellon University, Pittsburgh PA, 1991. Also published as CMU Technical Report CMU-CS-91-126.
54. E. Oja. A simplified neuron model as a principal component analyser. *Journal of Mathematical Biology*, 15:267–273, 1982.
55. E. Oja. Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1(1):61–68, 1989.
56. E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5:927–935, 1992.
57. E. Oja and J. Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106:69–84, 1985.
58. E. Oja, H. Ogawa, and J. Wangviwattana. PCA in fully parallel neural networks. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks, 2*, pages 199–202, Amsterdam, 1992. North-Holland.
59. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, California: Morgan Kaufmann, 1988.
60. B. A. Pearlmutter and G. E. Hinton. G-maximization: An unsupervised learning procedure for discovering regularities. In J. S. Denker, editor, *Neural Networks for Computing: American Institute of Physics Conference Proceedings 151*, pages 333–338, 1986.
61. C. Peterson and J. R. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.
62. C. Peterson and E. Hartman. Explorations of the mean field theory learning algorithm. *Neural Networks*, 2:475, 1989.
63. M. D. Plumbley. Efficient information transfer and anti-Hebbian neural networks. *Neural Networks*, 6(6):823–833, 1993.
64. M. D. Plumbley. A Hebbian/anti-Hebbian network which optimizes information capacity by orthonormalizing the principal subspace. In *Proceedings of the IEE Artificial Neural Networks Conference, ANN-93*, pages 86–90, Brighton, UK, May 1993.
65. M. D. Plumbley and F. Fallside. An information-theoretic approach to unsupervised connectionist models. In David Touretzky, Geoffrey Hinton, and Terrence Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 239–245. Morgan-Kaufmann, San Mateo, CA, 1988.
66. J. Rubner and P. Tavan. A self-organizing network for principal component analysis. *Europhysics Letters*, 10:693–698, 1989.
67. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT Press, Cambridge, MA, 1986.
68. D. E. Rumelhart and D. Zipser. Competitive learning. *Cognitive Science*, 9:75–112, 1985.
69. T. D. Sanger. Optimal unsupervised learning in a single-layer feedforward neural network. *Neural Networks*, 2:459–473, 1989.
70. E. Saund. Dimensionality-reduction using connectionist networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(3):304–314, 1989.
71. J. Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4:863–879, 1992.
72. J. Schmidhuber. Learning unambiguous reduced sequence decriptions. In *Advances in Neural Information Processing Systems 4*, pages 291–298. Morgan Kaufmann, 1992.
73. N. N. Schraudolph and T. J. Sejnowski. Competitive anti-hebbian learning of invariants. In *Advances in Neural Information Processing Systems 4*, pages 1017–1024. Morgan Kaufmann, 1992.
74. C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423,623–656, 1948.
75. A. Ukrainec and S. Haykin. Application of unsupervised neural networks to the enhancement of polarization targets in dual-polarized radar images. In *IEEE Canadian Confrernce on Elecrtical and Computer Engineering*, 1991.
76. C. von der Malsburg. Self-organization of orientation sensitive cells in striate cortex. *Kybernetik*, 14:85–100, 1973.
77. S. Watanabe. *Pattern Recognition: Human and Mechanical.* John Wiley & Sons, New York, 1985.
78. A. S. Weigend, B. A. Huberman, and D. E. Rumelhart. Predicting the future: A connectionist approach. *International Journal of Neural Systems*, 1:193-209, 1990.
79. R. J. Williams. Feature discovery through error-correction learning. ICS Report 8501, Institute of Cognitive Science, University of California, San Diego, 1985.
80. R. Zemel and G. E. Hinton. Developing topographic representations by minimizing description length. In J. D. Cowan, G. Tesauro and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 11–18. Morgan Kaufmann, 1994.
81. R. S. Zemel and G. E. Hinton. Discovering viewpoint-invariant relationships that characterize objects. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances In Neural Information Processing Systems 3*, pages 299–305. Morgan Kaufmann Publishers, 1991.

Suzanna Becker is an assistant professor of Psychology and is Programme Co-ordinator for the Honours Degree in Neural Computation at McMaster University, Hamilton, Ontario. Her research interests include unsupervised learning algorithms, information theory, statistical mixture models, and models of cortical plasticity, perceptual development, and human memory.

Dr. Becker received the BA degree (1982) from Queen's University in Psychology, the MSc degree (1985) from Queen's University in Computer Science, and the PhD degree (1992) from the University of Toronto in Computer Science.

Mark Plumbley is a Lecturer in the Department of Electronic and Electrical Engineering at King's College London. He is Deputy Director of the EC-funded "NEuroNet" Network of Excellence in Neural Networks, and Treasurer of the European Neural Network Society. His research interests include the use of information theory in neural networks, unsupervised learning, perceptual systems and genetic algorithms.

Dr. Plumbley received the BA degree (1984) from the University of Cambridge (Churchill College) in Electrical Science Tripos and the PhD (1991) from the University of Cambridge Department of Engineering in Neural Networks.