International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 17, No. 8 (2003) 1447–1466

World Scientific
www.worldscientific.com

CONNECTIONIST TECHNIQUES FOR THE IDENTIFICATION AND SUPPRESSION OF INTERFERING UNDERLYING FACTORS

EMILIO CORCHADO*
Departamento de Ingeniería Civil,
The University of Burgos, Burgos, Spain
escorchado@ubu.es

COLIN FYFE
**Applied Computational Intelligence Research Unit,*
The University of Paisley, Paisley, Scotland
colin.fyfe@paisley.ac.uk

We consider the difficult problem of identification of independent causes from a mixture of them when these causes interfere with one another in a particular manner: those considered are visual inputs to a neural network system which are created by independent underlying causes which may occlude each other. The prototypical problem in this area is a mixture of horizontal and vertical bars in which each horizontal bar interferes with the representation of each vertical bar and vice versa. Previous researchers have developed artificial neural networks which can identify the individual causes; we seek to go further in that we create artificial neural networks which identify all the horizontal bars from only such a mixture. This task is a necessary precursor to the development of the concept of "horizontal" or "vertical".

Keywords: Negative feedback network; ε-insensitive Hebbian learning rule; rectified Gaussian distribution; maximum/minimum likelihood Hebbian learning.

1. Introduction

Connectionist systems have achieved great success in the last fifteen years tackling problems such as optimization, regression, data compression and so on. However, one of the disadvantages of most connectionist systems is that they tend to be black boxes: they provide an answer but it is often impossible to state how the answer was arrived at because of the complexity of the system. One exception is to be found in those networks which perform data analysis such as Factor Analysis.[4,9,14,19,22]

The standard problem on which Factor Analysis type networks have been used is the "bars problem". This problem is an abstraction of one which is met by every living thing that extracts information from its visual environment: how do we reliably identify single objects from groups of objects which interfere with one

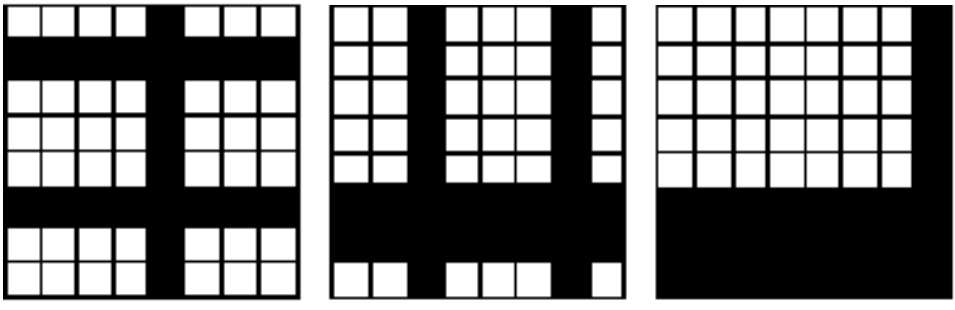


Fig. 1. Sample input bar patterns.

another through occlusion? An example of the bars problem is shown in Fig. 1. We wish the network to identify the individual bars when the input data is composed of random mixtures of the bars. Thus each set of weights in the artificial neural network will correspond to one bar only. Note that the interference which every horizontal bar causes to every vertical bar must be ignored by the network.

Several neural networks[4,9,14,19,22] have shown such features.

In this paper we wish to go further: we wish to investigate ways in which the concepts of "horizontal" and "vertical" may be identified (as they are in all humans) from only this mixture of bars. Thus, in this paper, we will create networks which, trained only on mixtures of horizontal and vertical bars, will actually self-organize so that only all the horizontal bars will be identified or only the vertical bars will be identified. This is clearly a necessary first stage on the road to creating the concepts of horizontal and vertical. This is an important stage beyond that performed by other researchers and we will show how to do this in three different ways.

The remainder of the paper is organized as follows. Section 2 discusses the negative feedback network which is central to the paper; Sec. 3 introduces the Rectified Gaussian Distribution from which lateral connections are derived and illustrates on artificial data, the effect of changing the strength of the lateral connections; Sec. 4 discusses a modification of the standard Hebbian learning rule; this modification is combined with the lateral connections in Sec. 5 where we present results on artificial data and a theoretical discussion of convergence; in Sec. 6, we discuss an extension to the modified Hebbian rule which sets the scene for our results on real data in Sec. 7.

2. Background

In this paper we present different methods which are suitable for the identification and suppression of underlying related factors. All these methods are based on the application of a negative feedback network which was first shown to be capable of performing Principal Component Analysis.

Principal Component Analysis (PCA) is a standard statistical technique for compressing data; it can be shown to give the best linear compression of the data

in terms of least mean square error. There are several artificial neural networks which have been shown to perform PCA e.g. Refs. 17 and 18. We shall be most interested in a negative feedback implementation.[10]

The basic PCA network[10] is described by Eqs. (1)–(3). Let us have an N-dimensional input vector at time t, $\mathbf{x}(t)$, and an M-dimensional output vector, $\mathbf{y}$, with W_{ij} being the weight linking input j to output i. η is a learning rate. Then the activation passing and learning is described by

$$\text{Feedforward :} \qquad y_i = \sum_{j=1}^{N} W_{ij} x_j, \quad \forall\, i \tag{1}$$

$$\text{Feedback :} \qquad e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i \tag{2}$$

$$\text{Change weights :} \qquad \Delta W_{ij} = \eta e_j y_i \,. \tag{3}$$

We can readily show that this algorithm is equivalent to Oja's Subspace Algorithm[17]:

$$\Delta W_{ij} = \eta e_j y_i = \eta \left(x_j - \sum_k W_{kj} y_k \right) y_i \tag{4}$$

and so this network not only causes convergence of the weights but causes the weights to converge to span the subspace of the Principal Components of the input data. We might ask then why we should be interested in the negative feedback formulation rather than the formulation (4) in which the weight change directly uses negative feedback. The answer, as we shall see in following sections, is that the explicit formation of residuals (2) allows us to consider probability density functions of the residuals in a way which would not be brought to mind if we use (4).

References 6 and 12 show that when positivity constraints are implemented in this negative feedback network, the resulting network performs an approximation to Factor Analysis. That is, either ensuring that the weights never become negative or ensuring that the outputs are always non-negative means that the resulting network extracts the underlying causes from a data set.

Thus this network will identify the individual bars in the bars dataset, something that the non-rectified PCA network cannot do.

3. The Rectified Gaussian Distribution

The Rectified Gaussian Distribution (RGD) can be used to form a relationship between the output neurons of a network so as to achieve, for example, a topographical ordering of factors.

3.1. *Definition*

The Rectified Gaussian Distribution is a modification of the standard Gaussian distribution in which the variables are constrained to be non-negative, enabling the use of non-convex energy functions.

The multivariate normal distribution can be defined in terms of an energy or cost function in that, if realized samples are taken far from the distribution's mean, they will be deemed to have high energy and this will be equated to low probability. More formally, Ref. 20 defined the standard Gaussian distribution by:

$$p(\mathbf{y}) = Z^{-1} e^{-\beta E(\mathbf{y})}\,, \tag{5}$$

$$E(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T A\mathbf{y} - \mathbf{b}^T\mathbf{y}\,. \tag{6}$$

The quadratic energy function $E(\mathbf{y})$ is defined by the vector $\mathbf{b}$ and the symmetric matrix A.

The parameter $\beta = 1/T$ is an inverse temperature. Lowering the temperature concentrates the distribution at the minimum of the energy function. One advantage of this formalization is that it allows us to visualize regions of high or low probability in terms of energy and hence to view movement to low energy regions as movement to regions of high probability. The factor Z normalizes the integral of $p(\mathbf{y})$ to unity.

3.2. *Mode-finding*

Note that the modes of the Rectified Gaussian are the minima of the energy function, subject to non-negativity constraints. The modes of the distribution characterize much of its behavior at low temperature. Finding the modes of a Rectified Gaussian is a problem in quadratic programming. However we will use what is probably the simplest algorithm, the projected gradient method, consisting of a gradient step followed by a rectification:

$$y_i(t+1) = [y_i(t) + \tau(b - Ay)]^+ \tag{7}$$

where the rectification $[\ ]^+$ is necessary to ensure that the y-values keep to the positive quadrant. Note that this rectification was one of the methods discussed earlier of transforming the negative feedback Principal Components Analysis network to a Factor Analysis network. If the step size τ is chosen correctly, this algorithm can provably be shown to converge to a stationary point of the energy function.[2] In practice, this stationary point is generally a local minimum.

The mode of the distribution can be approached by gradient descent on the derivative of the energy function with respect to $\mathbf{y}$. This is:

$$\Delta_{\mathbf{y}} \propto -\frac{\partial E}{\partial \mathbf{y}} = -(A_{\mathbf{y}} - \mathbf{b}) = \mathbf{b} - A_{\mathbf{y}}\,; \tag{8}$$

which is used as in (7).

3.3. *The cooperative distribution*

Using different values for A and $\mathbf{b}$ in Eq. (8), different regimes are defined. This research focuses on the use of the cooperative distribution which in the case of N variables is defined by:

$$A_{ij} = \delta_{ij} + \frac{1}{N} - \frac{4}{N}\cos\left(\frac{2\pi}{N}(i-j)\right) \tag{9}$$

$$b_i = 1 \tag{10}$$

where δ_{ij} is the Kronecker delta.

To speed learning up, the matrix A can be simplified[6] to:

$$A_{ij} = (\delta_{ij} - \cos(2\pi(i-j)/N)) \tag{11}$$

and is shown diagrammatically in Fig. 2. The matrix A is used to modify the response to the data based on the relation between the distances of the outputs. The outputs are thought of as located on a ring ("wraparound").

We use the standard negative feedback network but now with a lateral connection (which acts after the feed forward but before the feedback). It takes the form:

$$y_i(t+1) = [y_i(t) + \tau(b - Ay)]^+ \tag{12}$$

where the parameter τ represents the strength of the lateral connections.

Therefore, we are moving to the mode of the distribution.

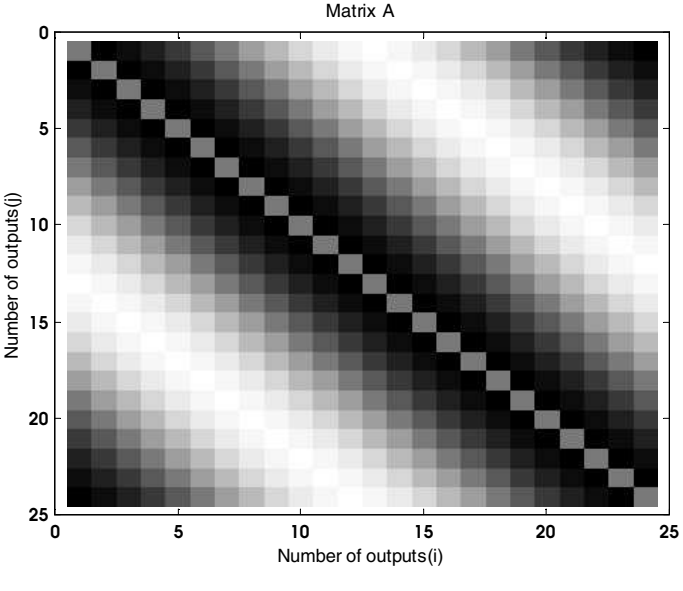


Fig. 2. A matrix for the Rectified Gaussian network with 24 outputs. Black squares are negative, white are positive and the shading in each square is proportional to the weight size.

This is called the Cooperative distribution because clusters of neurons, which are close, are activated together in response to some particular inputs.

The algorithm encourages correlations between nearby outputs and causes neurons to excite other nearby neurons. The network's other operation is the standard negative feedback operation.

3.4. *Experiments*

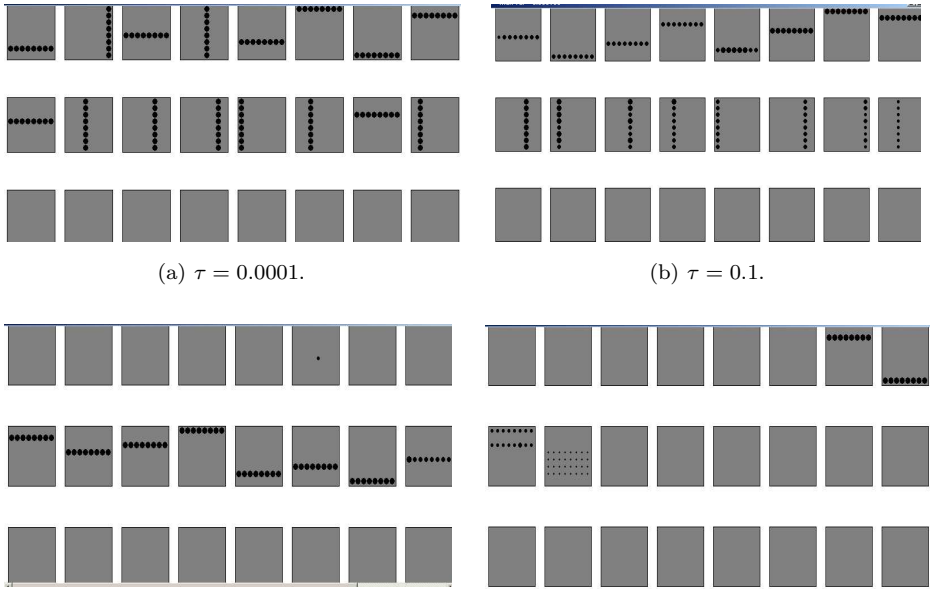
In this section, we discuss only experiments with artificial data. We show the results with real data[15] in Sec. 7.

3.4.1. *Effect of changing τ with artificial input data*

In this section we investigate the effect of changing the parameter τ, which is the strength of the lateral connections between the outputs. We use the Negative Feedback network with noise added in a graduated manner across the outputs, with the first output having the least noise added, the last having the most. The addition of noise in this way forces the coding of the features to shift across the output space and improves the coding performance of the network.[12] To show the effect of this parameter for both sorts of learning all the parameters have been kept steady except τ.

It has been shown[7] that by altering the strength of the lateral connection parameter we affect the ability of the network to "gather" features together on the outputs. With a low value of τ, we achieve a coding of both horizontal and vertical bars around a mode as predicted [Fig. 3(a)]. As we start to increase the value of τ, the weak correlations between horizontal and vertical bars begin to have an impact on the learning. As the strength of the lateral connections becomes stronger the bars are still learned around a mode but now orientations start to separate [Fig. 3(b)]. We have achieved a separation between the two different orientations. This is a remarkable demonstration since all data presentations to the network are of a mixture of horizontal and vertical bars.

Increasing the value of further forces the network to learn only one orientation of bar [Fig. 3(c)], however if the lateral connections are too strong then the coding of the bars may be squashed into an area of the output space that is too small for all bars to be coded individually [Fig. 3(d)]. The reason why one orientation of bar is suppressed [Fig. 3(c)] is due to the pixel overlap between different orientations of bars; if the lateral excitation between the output neurons is strong enough then a single output neuron may be able to switch its preference from a horizontal bar to a vertical one. For example, if, at an early point in the learning process, all bars have been learned and one single vertical bar is shown to the network then we will have two outputs (at least) responding to this pattern. The first is strong because it is associated with that particular bar, and the second because the bar that it usually responds to has a pixel overlap with the first. Due to the strong positive lateral connections this second output response is increased significantly due to the correlation with the first, and so the Hebbian-style update of the weights causes



(a) $\tau = 0.0001$. (b) $\tau = 0.1$.

(c) $\tau = 0.4$. (d) $\tau = 0.83$.

Fig. 3. Weight vectors obtained for four different values of the lateral connections parameter τ using To-Mode Learning.

this output to develop a preference for coding the same bar as the other output. As learning continues there are now two outputs learning the same bar but because these outputs are very strongly correlated, eventually only one output will take responsibility for this bar. In this way one orientation of bar may be eliminated from the coding — this may be horizontal or vertical.

4. ε-Insensitive Hebbian Learning Rule

In this section, we review an extension of the PCA network which has been derived to be optimal for a specific probability density function. We note that this probability density function is one of a family of pdfs and we will investigate later on the learning rules formed in order to be optimal for several members of this family.

4.1. *The cost function*

It has been shown[23] that the nonlinear PCA rule

$$\Delta W_{ij} = \eta\left(x_j f(y_i) - f(y_i)\sum_k W_{kj} f(y_k)\right) \tag{13}$$

can be derived as an approximation to the best nonlinear compression of the data.

We start with a cost function

$$J(W) = 1^T E\{(\mathbf{x} - W f(W^T \mathbf{x}))^2\} \tag{14}$$

which we minimize to get the rule (13). Reference 13 used the residual in the linear version of (14) to define a cost function of the residual

$$J = f_1(e) = f_1(\mathbf{x} - W\mathbf{y}) \tag{15}$$

where $f_1 = \|\cdot\|^2$ is the (squared) Euclidean norm in the standard linear or nonlinear PCA rule. With this choice of $f_1(\ )$, the cost function is minimized with respect to any set of samples from the data set on the assumption that the residuals are chosen independently and identically distributed from a standard Gaussian distribution.[3]

The minimization of J is equivalent to minimizing the negative log probability of the residual, $\mathbf{e}$.

Thus if

$$p(\mathbf{e}) = \frac{1}{Z}\exp(-\mathbf{e}^2) \tag{16}$$

then we can denote a general cost function associated with this network as

$$J = -\log p(\mathbf{e}) = (\mathbf{e})^2 + K \tag{17}$$

where K is a constant. Therefore performing gradient descent on J, we have

$$\Delta W \propto -\frac{\partial J}{\partial W} = -\frac{\partial J}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial W} \approx y(2\mathbf{e})^T\,. \tag{18}$$

In general,[21] the minimization of such a cost function may be the reason why the probability of the residuals is greater dependent on the pdf of the residuals. Thus if the probability density function of the residuals is known, this knowledge could be used to determine the optimal cost function.

Reference 13 investigated this with the (one-dimensional) function:

$$p(\mathbf{e}) = \frac{1}{2+\varepsilon}\exp(-|\mathbf{e}|_\varepsilon) \tag{19}$$

where

$$|e|_\varepsilon = \begin{cases} 0 & \forall\, |\mathbf{e}| < \varepsilon \\ |\mathbf{e}| - \varepsilon & \text{otherwise} \end{cases} \tag{20}$$

with ε being a small scalar ≥ 0.

Reference 13 described this in terms of noise in the data set. However we feel that it is more appropriate to state that, with this model of the pdf of the residual, the optimal $f_1(\ )$ function is the ε-insensitive cost function:

$$f_1(\mathbf{e}) = |\mathbf{e}|_\varepsilon\,. \tag{21}$$

In the case of the negative feedback network, the learning rule is

$$\Delta W \propto -\frac{\partial J}{\partial W} = -\frac{\partial f_1(\mathbf{e})}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial W} \tag{22}$$

which gives

$$\Delta W_{ij} = \begin{cases} 0 & \text{if } |e_j| < \varepsilon \\ \eta y(\text{sign}(e)) & \text{otherwise}\,. \end{cases} \tag{23}$$

The difference with the common Hebbian learning rule is that the sign of the residual is used instead of the value. Because this learning rule is insensitive to the magnitude of the input vectors $\mathbf{x}$, the rule is less sensitive to outliers than the usual rule based on mean squared error.

This change from viewing the difference after feedback as simply a residual rather than an error permits us to consider a family of cost functions each member of which is optimal for a particular probability density function associated with the residual.

5. Combining ε-Insensitive Hebbian Learning Rule and Lateral Connections

5.1. *Introduction*

We have already introduced lateral connections to the basic factor analysis network; these were derived from gradient ascent/descent on a pdf. In Sec. 4, we introduce the ε-Insensitive Hebbian Learning Rule which was derived from gradient ascent on a pdf. The question which obviously arises is whether and how these techniques can be combined. Therefore, we apply the ε-insensitive Hebbian learning rule and the use of lateral connections derived from the Rectified Gaussian distribution (RGD)[20] to our standard data sets.

5.2. *Results using the bars dataset*

In all experiments, the number of iterations is 50,000 and the learning rate is 0.01. During the first set of experiments we study the effect of the ε-insensitive Hebbian learning rule alone and then when combined with the application of lateral connections, derived from the RGD, on the outputs of the network.

The use of the ε-insensitive Hebbian learning rule allows the identification of all the independent causes (Fig. 4). This figure shows the weights in a trained network on an 8*8 grid (i.e. 16 bars). Each row of the diagram shows the learned weight vector in one output neuron — horizontal lines are found by eight continuous non-zero weights, vertical lines are by eight non-zero weights each separated from its neighbours by seven zero-magnitude weights.

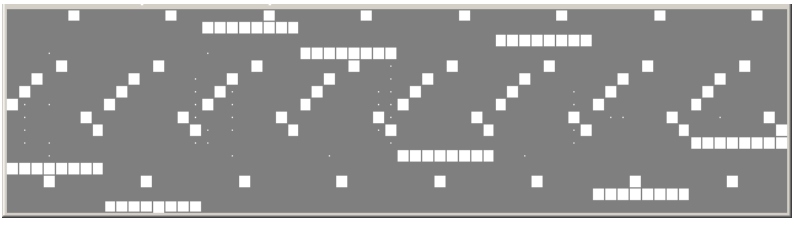


Fig. 4. Each row represents the weights in one output neuron. The eight horizontal and eight vertical lines have been found. The values used here were: number of outputs = 16 and $\varepsilon = 0.3$.

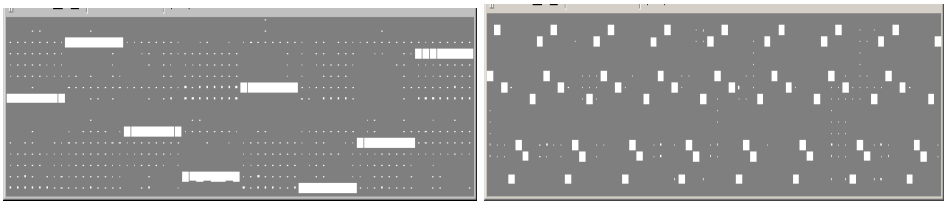


Fig. 5. The application of lateral connections allows the suppression of one orientation when the value of epsilon is set to a low value. The eight horizontal (left) or vertical (right) bars are found. These experiments had 16 outputs, and the strength of the lateral connections, $\tau = 0.2$; $\varepsilon = 0.05$.

The action of lateral connections combined with the ε-insensitive Hebbian learning rule allows the network to find all bars with a large enough value of ε (e.g. $\varepsilon = 0.5$) or just one orientation of bars (Fig. 5, for which $\varepsilon = 0.05$). Both situations, the identification of all bars or the suppression of one orientation, can be achieved by changing the value of ε and keeping the rest of the parameter values constant.

So when a specific optimal value of the lateral connections is found, changing ε allows the network to identify all the bars or suppresses one orientation of bars.

If we keep all the other parameters constant and we only change the value of the strength of the lateral connections (the τ parameter), the network will, as before, find one orientation of bar.

5.3. *Theoretical discussion*

The lateral weights increase interaction between bars of the same orientation.

(1) With relatively large values of ε, the system can stabilize to make the value of the ε-insensitive cost function $J_\varepsilon = |x - Wy|_\varepsilon$ equal to 0. When a bar is present, the weights stabilize such that (e.g. when $\varepsilon = 0.3$), the residual in an on-pixel is just less than 0.3 and that of an off-pixel goes to 0.1. Both of these residuals are within the ε-insensitive region and so $J_\varepsilon = 0$.
(2) As ε decreases, the system becomes over-constrained and it is not possible for J_ε to reach 0. Typically, with $\varepsilon \leq 0.05$, one orientation of bars comes to dominate — we find all n horizontal or all n vertical bars with none of the other orientation. We may call this the Maximum Non-interfering basis; note that it spans a subspace of the original space only; it is an incomplete basis for the data set.

Let horizontal bar h be on and let the neuron y_h have learned to recognize the bar. If neurons $y_{v_1}, \ldots y_{v_n}$ have learned the vertical bars, then each of them will be responding to the single pixel overlap with h. Now the network tends to a symmetrical response in that each weight tends to the same value. Let this value be C. Then neuron

$$y_h = \sum_j W_{hj} x_j = \sum_j W_{hj} = nC\,. \tag{24}$$

Similarly:

$$y_{v1} = y_{v2} = \cdots y_{vn} = C\,. \tag{25}$$

The lateral connections also have an effect at y_h

$$\begin{aligned} y_h &= y_h + \tau(1 - A\mathbf{y}) \\ &= nC + \tau(1 - A_{v_{1h}} y_{v_1} - \cdots - A_{v_{nh}} y_{v_n}) \\ &= nC + \tau(1 - A_{v_{1h}} C - \cdots - A_{v_{nh}} C) \end{aligned} \tag{26}$$

while at the vertical bar — responding neurons, we have

$$\begin{aligned} y_{v_i} &= y_{v_i} + \tau(1 - A\mathbf{y}) \\ &= C + \tau(1 - A_{v_1 v_i} C - \cdots - A_{v_n v_i} C - A_{hv_i} nC) \end{aligned} \tag{27}$$

where we have assumed that $A_{v_i v_i} = 0$.

Then at neuron y_{h_j}, $h_j \neq h$ the feedforward value $y_{h_j} = 0$, but the lateral connections introduce an interaction.

Then at neuron y_{h_j}:

$$\begin{aligned} y_{h_j} &= y_{h_j} + \tau(1 - Ay) \\ &= 0 + \tau(1 - A_{h_j h}(nC) - A_{h_j v_1}(C) - \cdots - A_{h_j v_n}(C))\,. \end{aligned} \tag{28}$$

Thus all output neurons will tend to some extent to be firing after the lateral interaction has taken place.

Now consider the feedback to a pixel, p, which is not on horizontal bar h.

Let it be a member of horizontal bar h_j and vertical bar V_i.

Then the error is:

$$\begin{aligned} e_p = x_p = x(h_j, v_i) &= 0 - W_{h_j p} y_{h_i} - W_{v_i p} y_{v_i} \\ &= -C\tau(1 - A_{h_j h} nC - A_{h_j v_1} C - A_{h_j v_2} C - \cdots - A_{h_j v_n} C) \\ &\quad - C(C + \tau(1 - A_{v_1 v_i} C - \cdots - A_{v_n v_i} C - \cdots - A_{hv_i} nC)) \\ &= C^2\tau(A_{h_j h} n + A_{h_j v_1} + A_{h_j v_2} + \cdots + A_{h_j v_n} \\ &\quad + A_{v_1 v_i} + \cdots + A_{v_n v_i} + A_{hv_i} n) - C^2 - 2C\tau\,. \end{aligned} \tag{29}$$

If ε is large, this error can be easily accommodated within the ε-insensitive region since $|C| < 1, |\tau| < 1, |A_{ij}| < 1,\ \forall\, ij$. However as ε decreases this become impossible and $|e_p|_\varepsilon > \varepsilon$ and so learning will take place.

The assumption of uniformity (all weights having a value C) is true only for a trained network. Training is a stochastic process depending on the bar(s) chosen as inputs at any one time and the initial random values of the weights. In practice, some neurons will initially have larger weights and hence fire more strongly than others. Let y_h now be such a strongly firing neuron such that $y_h > y_{v_i},\ \forall_i$. This

effect is also plausible in the following since each vertical neuron is responding to only one pixel of horizontal bar h.

Then after the lateral interaction:
$y_h \sim y_h$ i.e. y_n is little changed (it could actually have grown by τ)

$$y_{v_i} \sim y_{v_i} + \tau(1 - A_{hv_i} y_h)$$
$$y_{h_i} \sim \tau(1 - A_{h_j h} y_h), \quad h_j \neq h. \tag{30}$$

Let x_p be an input not on bar h. Then

$$\begin{aligned} e_p = x(h_j, v_i) &= 0 - W_{h_j p} y_{h_j} - W_{v_i p} y_{v_i} \\ &= -w_{h_j p}(\tau(1 - A_{h_j h} y_h)) - W_{v_i p}(y_{v_i}(1 - A_{hvi} y_h)). \end{aligned} \tag{31}$$

Let $y_{v_i} = y_{v_i} + \tau(1 - A_{hv_i} y_h) > 0$ and $y_{h_i} = \tau(1 - A_{h_j h} y_h) > 0$ which will be true at the start of training when all weights are fairly small. Now the weights are constrained to be non-negative and so $e_p < 0$. The learning rule gives $\Delta W_{v_i p} = \eta(-1) y_{v_i}$ i.e. the weights will decrease in value and so will tend to turn off this weight. Now the same argument also holds for y_{h_j} but crucially y_{h_j} is 0 before the lateral connection and continues to be smaller after lateral interaction and so $\Delta W_{p h_j} = \eta(-1) y_{h_j} < \Delta W_{p v_i}$, i.e. the horizontal neuron turns off the vertical neurons faster than it turns off other horizontal neurons.

Note that as y_h grows, each of y_{v_i} and y_{h_j} may become negative but because of the negative feedback, e_p will then become positive which will make $\Delta W_{p v_i}$ and $\Delta W_{p h_i}$ negative, again forcing the weights to decrement.

Another point to note is the interaction between bars of the same orientation. Recall that without the lateral interaction stage, neurons responding to bars of the same orientation had no effect on one another. But now each tries to turn off the other by an amount proportional to $A_{v_i v_j}$ or $A_{h_i h_j}$. During training, it is typical for one bar (e.g. v_j) to be learned first. Because of the foregoing argument it tends to inhibit the network from responding to any other bar by an amount proportional to the corresponding A matrix. However, neurons which have A_{ij} negative will be in best position to respond and will respond best to other bars of the same orientation. Thus vertical bars tend to be learned spatially separated on the grid of $2n$ (or $3n$ or $4n$) outputs. This is a stable solution since the groups of vertical bars are in effect cooperating with one another.

6. General Maximum Likelihood Hebbian Learning

Now the ε-insensitive learning rule is clearly only one of a possible family of learning rules which are suggested by a family of distributions based on the exponential function. Let the residual after feedback have probability density function

$$p(\mathbf{e}) = \frac{1}{Z} \exp(-|\mathbf{e}|^p). \tag{32}$$

Then we can denote a general cost function associated with this network as

$$J = -\log p(\mathbf{e}) = |\mathbf{e}|^p + K \tag{33}$$

where K is a constant. Therefore performing gradient descent on J, we have

$$\Delta W \propto -\frac{\partial J}{\partial W} = -\frac{\partial J}{\partial \mathbf{e}} \frac{\partial \mathbf{e}}{\partial W} \approx y(p|\mathbf{e}|^{p-1}\ \text{sign}(\mathbf{e}))^T \tag{34}$$

where T denotes the transpose of a vector. We would expect that for leptokurtotic residuals (more kurtotic than a Gaussian distribution), values of $p < 2$ would be appropriate, while for platykurtotic residuals (less kurtotic than a Gaussian), values of $p > 2$ would be appropriate.

Therefore the network operation is:

$$\text{Feedforward}: \qquad y_i = \sum_{j=1}^{N} W_{ij} x_j, \quad \forall_i \tag{35}$$

$$\text{Feedback}: \qquad e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i \tag{36}$$

$$\text{Weight change}: \qquad \Delta W_{ij} = \eta \cdot y_i \,\text{sign}(e_j)|e_j|^p\,. \tag{37}$$

6.1. *Experiments*

We have performed experiments combining various Maximum Likelihood Hebbian learning rules, an ε-insensitive region and Lateral Connections.

We use a greater number of outputs, 20, than the number of bars, 16. The value of the lateral connection is 0.2, the value of ε is 0.06, the learning rate is 0.01 and the number of iterations is 50,000. We added no noise.

The suppression of one orientation of bar is achieved when $p = 1$, which corresponds to ε-Insensitive learning rule which is more appropriate for data sets which are kurtotic.

For $p = 2$, the network recognized the 16 bars in 16 outputs with the other 4 outputs having weights approximately 0. For p greater than 2, some of the bars are recognized by more than one weight vector.

In the case of $p = 1$, which is appropriate for positively kurtotic data, the residual is either a whole bar of the opposite orientation to that in the data presented or is 0. In other words, we get a lot of residuals which are close to zero and which are very large. This is a prescription for a kurtotic distribution and so only a single orientation is found by the network.

For p greater than 2, the network aims to maximize the probability that the residuals are from a distribution with negative kurtosis which could be thought of as a distribution which is rather more uniform (or even bimodal) compared with a Gaussian. In this situation, sometimes two neurons respond to one bar. So the coding is more spread out and the residuals are liable to be more uniform than would be the case when each bar is identified by a single neuron.

7. Application to Visual Data

Both architectures, the negative feedback network with lateral connections and the family of Maximum Likelihood Hebbian learning rules have been used with real visual data.[15]

Thus movement in these images could be due to camera panning, movement of the targets, zooming of the camera, etc. These movies have been produced in such a way that single-cause movements have been captured. For example, keeping the camera still and capturing the movement of objects in front of the camera, rotating the camera through 360° while capturing still scenes, and by holding the camera still and zooming in on stationary objects. The movie called "Trees" was shot by rotating the camera in the horizontal plane in a forest surrounded by trees.

The movies are converted into a set of equal sized sequential still images and each image transformed to a storage matrix with each pixel having a value in the range [−127, 128]. These images are sampled to produce the inputs to the network.

A random starting point is selected in the sequence of still images (see Fig. 6). We select as an input a square sample of size 12×12 pixels and then the corresponding square samples in exactly the same position in each of the next 11 still images in the sequence. This gives us a $12 \times 12 \times 12$ sample patch and all pixel values from this sample are presented to the network simultaneously as an input vector of length 1728. This process is repeated for every cycle of the learning algorithm.

These filters are local in time since, for instance, in neuron 1, we see zero weight values in the first three time slices and in the five last time slices, and only non-zero values in slices 4–7. They are local in space because in those non-zero filters, especially in the fifth and sixth slices, they are just values going from the top to the bottom only in the middle of the slices.

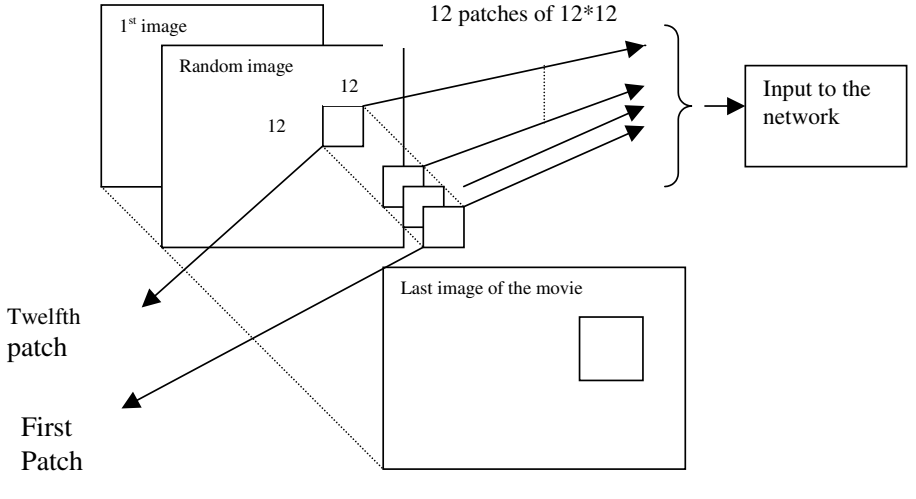


Fig. 6. A $12 \times 12 \times 12$ input vector of the network. Figure 7 shows the kind of filters that can be obtained using this data.

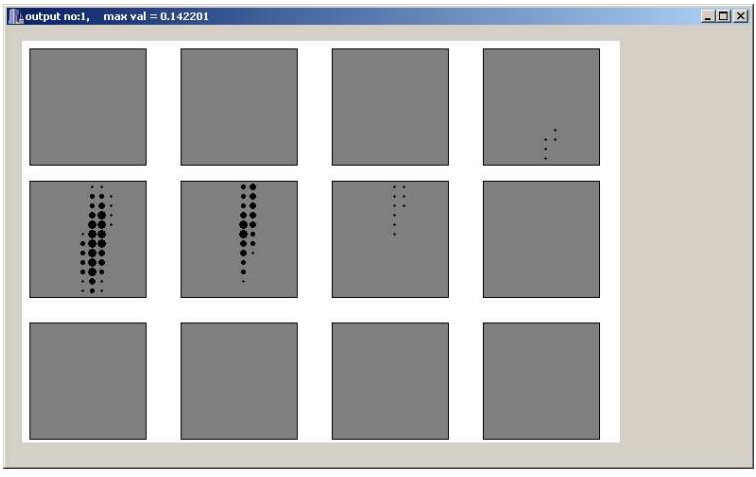


Fig. 7. This figure shows the weight vectors connected to an output neuron. The sizes of the weights are represented by the diameter of the small black circles (black represents a positive value). The square boxes within each rectangular window coincide with the weights connecting the output to an input sample from each of the 12 sequential image patches from the movie.

In the following set of images we show these filters for one of the movies called "Trees" and the action of the lateral connections. Figure 8 shows four consecutive weight vectors which again are each local in time and space. In addition, there is a global ordering to the filters. Weights in neuron 1 detect movement in time slices 3–5, weights in neuron 2 detect movement in time slices 5 and 6; weights in neuron 3 detect movement in time slices 6–10 and weights in neuron 4 detect movement in time slices 9–12.

This phenomenon may be related to the perception across the cortex of gradual movements of objects. This is the first time that such global organization has been created using such kind of input data.

We have also investigated the use of the family of Maximum Likelihood Hebbian rules on the video data set. In Fig. 9, we show the type of filters found when we use $p = 3$; in Fig. 10, we show the type of filters found when we use $p = 2$; in Fig. 11, we give an example of a filter found when $p = 1$; all networks used rectification of the weights.

Now (see Ref. 8), visual data is characterized by elements of positive kurtosis. For example, Ref. 1 show that edges in images tend to have positively kurtotic distributions. Therefore, our first interesting investigation is to find out what happens when our network is aiming to maximize the probability of residuals under a negatively kurtotic distribution. Now it is not possible, in general, to find negative kurtosis in the data set and so the network's best response is to remove any positive kurtosis in the data and have the residuals more close to Gaussian. Thus, we see (Fig. 9) that for values of $p > 2$, we achieve a very sparse response on the outputs.

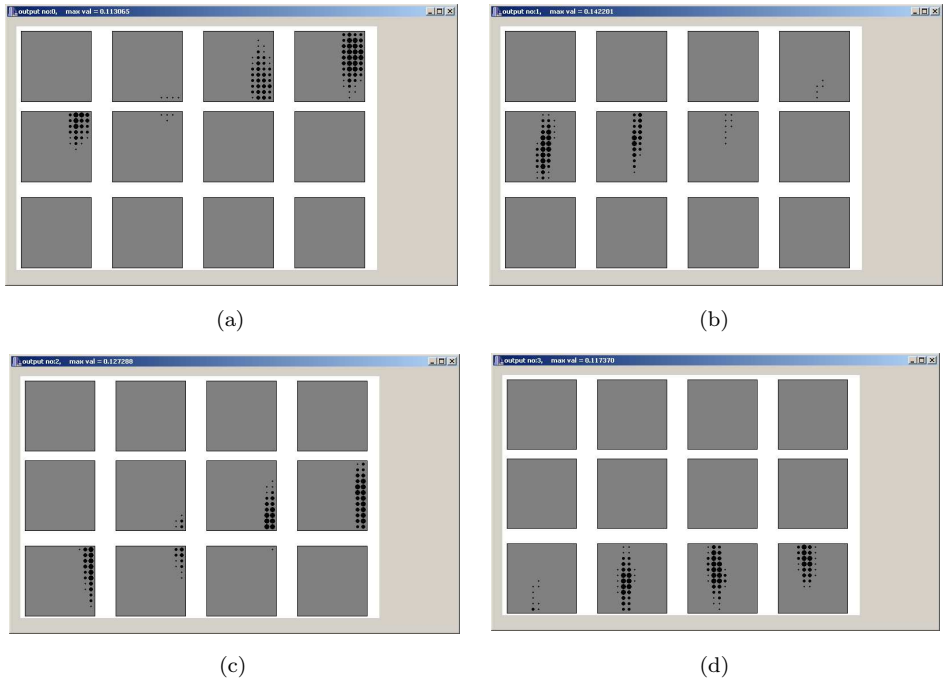


Fig. 8. The figure shows four weight vectors (from (a)–(d)) connected to four different outputs of the network. A rectification on the weight vectors and on the outputs is performed. Lateral connections are applied on the outputs of the network. Each rectangular box shows the weight vectors connected to a different output neuron.

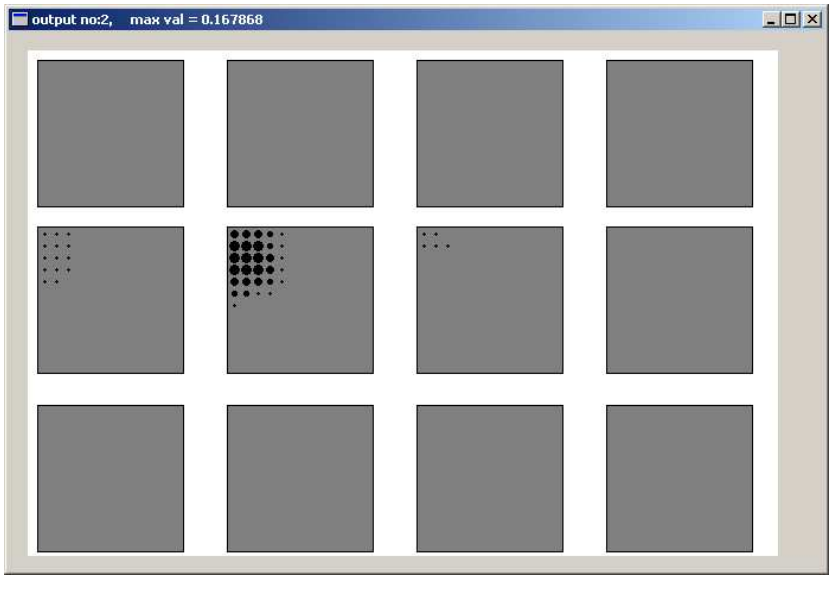


Fig. 9. The very local filter found with Maximum Likelihood Hebbian Learning with $p = 3$.

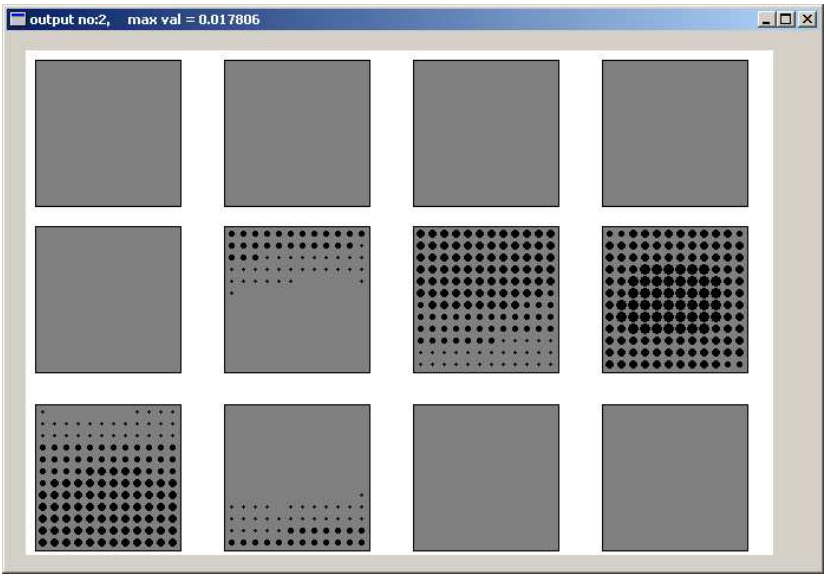


Fig. 10. The fairly local in time and space filters found by the Maximum Likelihood Hebbian learning with $p = 2$.

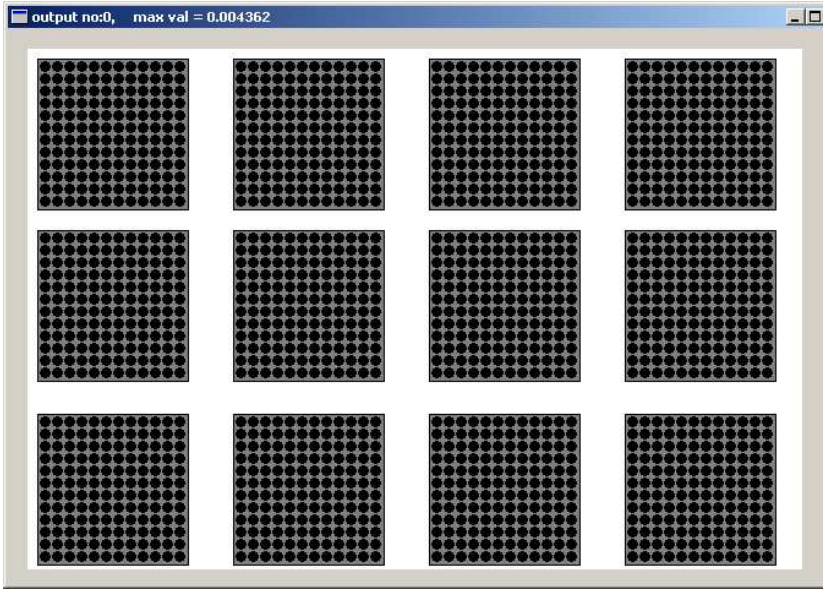


Fig. 11. The far from local (or informative) filter found with Maximum Likelihood Hebbian learning with $p = 1$.

Figure 10 is the standard network which has previously been analyzed and the results herein are compatible with those found earlier and are really only included for completeness in this section. Figure 11, on the other hand, is the result of searching for positively kurtotic residuals and here our network tells us that positive kurtosis is liable to be in any one part of the image as in any other part. However this kurtosis is liable to be tempered with sizeable quantities of Gaussian noise (possibly augmented by the sampling technique) and so the network simply attempts to remove this Gaussian noise from the whole image.

We conclude that local filters can be developed in an organism learning on visual data with learning rules of the type developed here only when using rules with $p \geq 2$.

8. Conclusions

In this paper, we have combined two connectionist techniques which extend a basic negative feedback network:

1. The first technique has been to create lateral connections at the output of the neural network. The magnitude of these lateral connections has been derived from gradient descent on the Cooperative Rectified Gaussian Distribution. This has ensured that there is both positive and negative reactions between output values and has led the activation settling to one of the modes of this distribution.
2. The second technique is an extension of Hebbian learning based on creating a model of the residuals and deriving a learning rule from this model to change the weights so that the residuals found in practice are most likely under this model.

Jointly these changes have enabled us to

- identify the underlying causes in an artificial data;
- suppress one orientation of bar.

The first of these has been performed before using other artificial neural networks but the second stage is the crucial stage introduced in this paper. In Ref. 4, we used a two-layer network to identify horizontal and vertical bars separately; however, in that network, the data set was very much simpler — "a random mixture of only one type of bar at a time (probability of any bar appearing is 1/6) to give the second layer the opportunity to categorize". In Ref. 5, we also separated the horizontal and vertical bars, however, the data set now comprised "bars appear[ing] as part of horizontally or vertically moving sequences". In the current work, our mixture is truly a random mixture of horizontal and vertical bars each of which appears in each mixture with a set probability independent of the other bars.

We consider this stage to be a precursor of the creation of the concept of horizontal or vertical in an animal which inhabits a mixed environment. We have performed this task in three different ways, by:

(a) Changing the strength of the lateral connections.

(b) Combining the action of the lateral connections and (ε-insensitive Hebbian learning rule.
(c) Using the appropriate member of the Maximum/Minimum Likelihood Hebbian Learning family.

We propose in future work to investigate whether these techniques can be applied to other data sets in which the interference from independent causes may be more complex than that which is exhibited herein.

References

1. H. B. Barlow and D. J. Tolhurst, "Why do you have edge detectors," *1992 Optical Society of America Annual Meeting*, 1992 Technical Digest series, Vol. 23, Albuquerque, NM, Optical Society of America, Washington.
2. D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1995.
3. C. M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.
4. D. Charles and C. Fyfe, "Modelling multiple cause structure using rectification constraints," *Network*: *Comput. Neural Syst.* **9** (1998) 167–182.
5. D. Charles, C. Fyfe, D. MacDonald and J. Koetsier, "Unsupervised neural networks for the identification of mimimum overcomplete basis in visual data," *Neurocomputing* **47** (2002) 119–143.
6. D. Charles, "Unsupervised artificial neural networks for the identification of multiple causes in data," Ph.D. Thesis, University of Paisley, 1999.
7. E. Corchado, D. Charles and C. Fyfe, "Rectified Gaussian distributions and the formation of local filters from video," *9th Eur. Symp. Artificial Neural Networks, ESANN'2001*, April 2001, pp. 397–402.
8. D. J. Field, "What is the goal of sensory coding?" *Neural Comput.* **6** (1994) 559–601.
9. P. Földiák, "Models of sensory coding," Ph.D. Thesis, University of Cambridge, 1992.
10. C. Fyfe, "PCA properties of interneurons," *From Neurobiology to Real World Computing*, *Proc. Int. Conf. Artificial Neural Networks*, *ICANN'93*, 1993, pp. 183–188.
11. C. Fyfe, "A neural network for PCA and beyond," *Neural Process. Lett.* **6** (1997) 33–41.
12. C. Fyfe and D. Charles, *Using Noise to Form a Minimal Overcomplete Basis*, *ICANN '99*, 1999, pp. 708–713.
13. C. Fyfe and D. MacDonald, "ε-insensitive Hebbian learning," *Neurocomputing* **47** (2002) 35–57.
14. Z. Ghahramani and G. E. Hinton, "Hierarchical nonlinear factor analysis and topographical maps," *Advances in Neural Information Processing Systems*, NIPS10, cd version nips10\0486.djvu, 1998.
15. Y. Han and C. Fyfe, "An investigation of video data using neural networks for factor analysis and principal component analysis," *Comput. Inform. Syst. J.* **8** (2001) 9–76.
16. A. Hyvärinien, "Complexity pursuit: separating interesting components from time series," *Neural Comput.* **13** (2001) 883–898.
17. E. Oja, "Neural networks, principal components and subspaces," *Int. J. Neural Syst.* **1** (1989) 61–68.

18. E. Oja, H. Ogawa and J. Wangviwattana, "Principal components analysis by homogeneous neural networks, Part 1: The weighted subspace criterion," *IEICE Trans. Inform. Syst.* **E75D** (1992) 366–375.
19. E. Saund, "A multiple-cause mixture model for unsupervised learning," *Neural Comput.* **7** (1995) 51–71.
20. H. S. Seung, N. D. Socci and D. D. Lee, "The rectified Gaussian distribution," *Ad. Neural Inform. Process. Syst.* **10** (1998) 350–356.
21. A. J. Smola and B. Scholkopf, "A tutorial on support vector regression," Technical Report NC2-TR-1998-030, NeuroCOLT2 Technical Report Series, October 1998.
22. R. H. White, "Competitive Hebbian learning: algorithm and demonstration," *Neural Networks* **5** (1992) 261–275.
23. L. Xu, "Least mean square error reconstruction for self-organizing nets," *Neural Networks* **6** (1993) 627–648.

E. Corchado is a full professor of computer science at the University of Burgos, Spain. He received his Ph.D. in computer science and his undergraduate degree in physics from the University of Salamanca (Spain). He is currently doing a Ph.D. on artificial neural networks at University of Paisley (UK) under the supervision of Prof. Colin Fyfe. He is a member of the Applied Computational Intelligence Research Unit at the University of Paisley, Scotland and director of the "Grupo de Inteligencia Computacional Aplicada-(GICAP)" at the University of Burgos, Spain.

His research interests include artificial neural networks, with a particular focus on exploratory projection pursuit, self-organising maps and kernel methods.

C. Fyfe, B.Sc. M.Sc. MEd. Ph.D. is a Personal Professor at the University of Paisley. He has published over 200 papers in the fields of artificial neural networks and genetic algorithms and has supervised 13 successful Ph.D. candidates in the last five years. He is on the Editorial Board of the *Int. J. Neural Systems* and the *Int. J. Knowledge-Based Intelligent Engineering Systems*.