Data Mining and Knowledge Discovery, 8, 203–225, 2004
© 2004 Kluwer Academic Publishers. Manufactured in The Netherlands.

Maximum and Minimum Likelihood Hebbian Learning for Exploratory Projection Pursuit

EMILIO CORCHADO emilio.corchado@paisley.ac.uk
DONALD MACDONALD donald.mcdonald@paisley.ac.uk
COLIN FYFE colin.fyfe@paisley.ac.uk
Applied Computational Intelligence Research Unit, The University of Paisley, Scotland

Editors: Fayyad, Mannila, Ramakrishnan

Received January 23, 2002; Revised May 29, 2002

Abstract. In this paper, we review an extension of the learning rules in a Principal Component Analysis 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 investigate the learning rules formed in order to be optimal for several members of this family. We show that, whereas we have previously (Lai et al., 2000; Fyfe and MacDonald, 2002) viewed the single member of the family as an extension of PCA, it is more appropriate to view the whole family of learning rules as methods of performing Exploratory Projection Pursuit. We illustrate this on both artificial and real data sets.

Keywords: exploratory projection pursuit, artificial neural networks

Introduction

In this paper, we investigate ways of extracting information from high dimensional data sets by projecting the data sets onto low dimensional (typically 2 dimensional) subspaces. We relate the artificial neural network methods we derive to statistical techniques developed for this purpose.

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. Oja (1989) and Oja et al. (1992). We shall be most interested in a negative feedback implementation (Fyfe, 1993).

The basic PCA network (Fyfe,1993) 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

Feedforward:

$$y_i = \sum_{j=1}^{N} W_{ij} x_j, \forall i \tag{1}$$

Feedback:

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

Change weights:

$$\Delta W_{ij} = \eta e_j y_i. \tag{3}$$

We can readily show that this algorithm is equivalent to Oja's Subspace Algorithm (Oja, 1989):

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

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 the next section, 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).

Exploratory Projection Pursuit (EPP) is a more recent statistical method aimed at solving the difficult problem of identifying structure in high dimensional data. It does this by projecting the data onto a low dimensional subspace in which we search for its structure by eye. However not all projections will reveal the data's structure equally well. We therefore define an index that measures how "interesting" a given projection is, and then represent the data in terms of projections that maximise that index.

The first step in our exploratory projection pursuit is to define which indices represent interesting directions. Now "interesting" structure is usually defined with respect to the fact that most projections of high-dimensional data onto arbitrary lines through most multi-dimensional data give almost Gaussian distributions (Diaconis and Freedman, 1984). Therefore if we wish to identify "interesting" features in data, we should look for those directions onto which the data-projections are as far from the Gaussian as possible.

Two simple measures of deviation from a Gaussian distribution are based on the higher order moments of the distribution. Skewness is based on the normalised third moment of the distribution and measures the deviation of the distribution from bilateral symmetry. Kurtosis is based on the normalised fourth moment of the distribution and measures the heaviness of the tails of a distribution. A bimodal distribution will often have a negative kurtosis and therefore negative kurtosis can signal that a particular distribution shows evidence of clustering.

Because a Gaussian distribution with mean a and variance x is no more or less interesting than a Gaussian distribution with mean b and variance y—indeed this second order structure can obscure higher order and more interesting structure—we remove such information from the data. This is known as “sphering”. That is, the raw data is translated till its mean is zero, projected onto the principal component directions and multiplied by the inverse of the square root of its eigenvalue to give data which has mean zero and is of unit variance in all directions.

We have previously implemented EPP using an artificial neural network (Fyfe and Baddeley, 1995); the method is essentially a non-linear modification of the negative feedback network. The network can be described by the following set of equations

$$s_i = \sum_{j=1}^{N} W_{ij} x_j \tag{5}$$

$$e_j = x_j - \sum_{k=1}^{M} W_{kj} s_k \tag{6}$$

$$r_i = f(s_i) \tag{7}$$

$$\Delta W_{ij} = \eta r_i e_j \tag{8}$$

where x_j is the sphered activation of the jth input neuron, s_i is the activation of the ith output neuron, W_{ij} is the weight between these two and r_i is the value of the function $f()$ on the ith output neuron.

It was shown in Karhunen and Joutsensalo (1994) that the use of a (non-linear) function $f()$ in Eq. (7) creates an algorithm to find those values of W which maximise that function whose derivative is $f()$ under the constraint that W is an orthonormal matrix. This was applied in Fyfe and Baddeley (1995) to the above network in the context of the network performing an Exploratory Projection Pursuit. Thus if we wish to find a direction which maximises the kurtosis of the distribution which is measured by s^4, we will use a function $f(s) \approx s^3$ in the algorithm. If we wish to find that direction with maximum skewness, we use a function $f(s) \approx s^2$ in the algorithm.

ε-Insensitive Hebbian learning

It has been shown (Xu, 1993) that the nonlinear PCA rule

$$\Delta W_{ij} = \eta\left(x_j f(y_i) - f(y_i) \sum_k W_{kj} f(y_k)\right) \tag{9}$$

can be derived as an approximation to the best non-linear compression of the data.

Thus we may start with a cost function

$$J(W) = 1^T E\{(\mathbf{x} - W f(W^T \mathbf{x}))^2\} \tag{10}$$

which we minimise to get the rule (9). Fyfe and MacDonald (2002) used the residual in the linear version of (10) to define a cost function of the residual

$$J = f_1(\mathbf{e}) = f_1(\mathbf{x} - W\mathbf{y}) \tag{11}$$

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 (Bishop, 1995).

We may show that the minimization of J is equivalent to minimizing the negative log probability of the residual, $\mathbf{e}$, if $\mathbf{e}$ is Gaussian.

Let

$$p(\mathbf{e}) = \frac{1}{Z}\exp(-\mathbf{e}^2). \tag{12}$$

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

$$J = E(-\log p(\mathbf{e})) = E((\mathbf{e})^2 + K) \tag{13}$$

where K is a constant. Therefore performing gradient descent on an instantaneous version of J (i.e. ignoring the expectation) we have

$$\Delta W \propto -\frac{\partial J}{\partial W} = -\frac{\partial J}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial W} \approx \mathbf{y}(2\mathbf{e})^T \tag{14}$$

where we have discarded a less important term (see Karhunen and Joutsensalo (1994) for details).

In general (Smola and Scholkopf, 1998), the minimisation of such a cost function may be thought to make the probability of the residuals 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.

Fyfe and MacDonald (2002) investigated this with the (one dimensional) function:

$$p(\mathbf{e}) = \frac{1}{2+\varepsilon}\exp(-|\mathbf{e}|_\varepsilon) \tag{15}$$

where

$$|e|_\varepsilon = \begin{cases} 0 & \forall |\mathbf{e}| < \varepsilon \\ |\mathbf{e}| - \varepsilon & \text{otherwise} \end{cases} \tag{16}$$

with ε being a small scalar ≥ 0.

Fyfe and MacDonald (2002) 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:

$$f_1(\mathbf{e}) = |\mathbf{e}|_\varepsilon. \tag{17}$$

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

$$\Delta W \propto -\frac{\partial J}{\partial W} = -\frac{\partial f_1(\mathbf{e})}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial W} \tag{18}$$

which gives:

$$\Delta W_{ij} = \begin{cases} 0 & \text{if} |e_j| < \varepsilon \\ \eta y(\text{sign}(\mathbf{e})) & \text{otherwise} \end{cases} \tag{19}$$

The difference with the common Hebb learning rule is that the sign of the residual is used instead the value of the residual. Because this learning rule is insensitive to the magnitude of the input vectors **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.

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 the family of exponential distributions.[1] Let the residual after feedback have probability density function

$$p(\mathbf{e}) = \frac{1}{Z}\exp(-|\mathbf{e}|^p). \tag{20}$$

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

$$J = E(-\log p(\mathbf{e})) = E(|\mathbf{e}|^p + K) \tag{21}$$

where K is a constant independent of W and the expectation is taken over the input data set. Therefore performing gradient descent on J we have

$$\Delta W \propto -\frac{\partial J}{\partial W}\bigg|_{W(t-1)} = -\frac{\partial J}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial W}\bigg|_{W(t-1)} \approx E\left\{\mathbf{y}(p|\mathbf{e}|^{p-1}\text{sign}(\mathbf{e}))^T\big|_{W(t-1)}\right\} \tag{22}$$

where T denotes the transpose of a vector and the operation of taking powers of the norm of **e** is on an elementwise basis as it is derived from a derivative of a scalar with respect to a vector.

Computing the mean of a function of a data set (or even the sample averages) can be tedious, and we also wish to cater for the situation in which samples keep arriving as we investigate the data set and so we derive an online learning algorithm. If the conditions of stochastic approximation (see Kashyap et al., 1994) are satisfied, we may approximate this with a difference equation. The function to be approximated is clearly sufficiently smooth and the learning rate can be designed to approximately satisfy $\eta_k \geq 0, \sum_k \eta_k = \infty, \sum_k \eta_k^2 < \infty$ and so we have the rule:

$$\Delta W_{ij} = \eta \cdot y_i \cdot \text{sign}(e_j)|e_j|^p.$$

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. Researchers from the community investigating Independent Component Analysis (Hyvärinen, 2001; Hyvärinen et al., 2002, p. 206) have shown that it is less important to get exactly the correct distribution when searching for a specific source than it is to get an approximately correct distribution i.e. all supergaussian signals can be retrieved using a generic leptokurtotic distribution and all subgaussian signals can be retrieved using a generic platykutotic distribution. Our experiments will tend to support this to some extent but we often find accuracy and speed of convergence are improved when we are accurate in our choice of p.

Therefore the network operation is:

Feedforward:

$$y_i = \sum_{j=1}^{N} W_{ij} x_j, \quad \forall_i \tag{23}$$

Feedback:

$$e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i \tag{24}$$

Weight change:

$$\Delta W_{ij} = \eta \cdot y_i \cdot \text{sign}(e_j)|e_j|^p \tag{25}$$

Fyfe and MacDonald (2002) described their rule as performing a type of PCA, but this is not strictly true since only the original (Oja) ordinary Hebbian rule actually performs PCA. It might be more appropriate to link this family of learning rules to Principal Factor Analysis since PFA makes an assumption about the noise in a data set and then removes the assumed noise from the covariance structure of the data before performing a PCA. We are doing something similar here in that we are basing our PCA-type rule on the assumed distribution of the residual. By maximising the likelihood of the residual with respect to the actual distribution, we are matching the learning rule to the pdf of the residual.

More importantly, we may also link the method to the standard statistical method of Exploratory Projection Pursuit: now the nature and quantification of the interestingness is in terms of how likely the residuals are under a particular model of the pdf of the residuals. In the results reported later, we also sphere the data before applying the learning method to the sphered data and show that with this method we may also find interesting structure in the data.

Connection to nonlinear PCA

Since the above rule has been derived as an extension to a basic PCA network, we might reasonably ask what connection it has to other rules derived from a similar perspective. Unarguably the most influential investigator of such rules has been Oja who (see Hyvärinen et al. (2002) for a recent review) derived the Nonlinear PCA rule from the criterion $J(W) = 1^T E\{(\mathbf{x} - Wf(W^T\mathbf{x}))^2\}$ where we have used our notation to make comparison easier. Hyvärinen et al. (2002) also derive their Independent Component Analysis rules from a Maximum Likelihood perspective but notice that the mean squared error is being used and also, in our neural network formulation, the nonlinearity is performed at the outputs before feedback. Oja then derives the rules $\Delta W = \eta f(W^T\mathbf{x}) \cdot \mathbf{e}^T$ (our notation). Note the differences between this rule and that investigated in this paper: the **e** in Oja's rule is formed from the feedback containing the nonlinear term whereas in the Maximum Likelihood rule investigated here, the **e** contains only feedback from a linear output. Our nonlinearity comes from a function then acting on the **e** as a whole. For completeness, we may also compare our previous Exploratory Projection Pursuit rule which may be written $\Delta W = \eta f(W^T\mathbf{x}) \cdot (\mathbf{x} - WW^T\mathbf{x})^T$ in which we see that the **e** term contains only linear feedback while the nonlinearity is only used in the first term of the learning rule. Thus the new rules are of a different type from those investigated previously.

A fixed point algorithm?

Given that Hyvärinen et al. (2002) have derived a Fixed Point Algorithm (known as FastICA) from Maximum Likelihood principles, we might reasonably ask whether a similar algorithm might be introduced here. The fixed point algorithm is derived by noting that, at a stable point of the gradient algorithm, the gradient must point in the direction of the weights since otherwise convergence would continue and we would not then be at a fixed point. In the current context, this would give us a rule

$$W \leftarrow E\left\{\mathbf{y}(p|\mathbf{e}|^{p-1}\text{sign}(\mathbf{e}))^T\big|_{W(t-1)}\right\}$$

Unfortunately, there is a difficulty with this method: the weights do converge very quickly (1 or 2 iterations typically gives over 99% accuracy) but the algorithm is not stable at this point. We are searching to maximise the likelihood that the residuals match some distribution; if the current values of the weights are removing most of the match of the data to this distribution, the algorithm simply finds the best projection it can from the *current residuals given by the current weights*. Thus learning continues though the magnitude of the new weights can be seen to be very much smaller than those used previously. Subsequently the previous optimal filter is found again and the program simply oscillates between finding the optimum and then a subspace orthogonal to that optimum. We have found that a reasonable compromise is one iteration of the fixed point algorithm, followed by gradient descent to fine tune the first estimate. However, the simulations we report on in this paper use solely gradient descent.

Experimental results

To illustrate our method, we follow Fyfe and Baddeley (1995) in creating artificial data sets, each of 10 dimensions. All results reported are based on a set of 10 simulations each with different initial conditions. It is our general finding that sphering is necessary to get the most accurate results presented below.

Artificial data set 1. In this data set, we have 9 independent leptokurtotic dimensions and one Gaussian dimension; this is almost the opposite of the standard EPP data sets described in Fyfe (1995) and is rather far from being a typical data set in that most projections onto its natural basis are interesting. However, since we wish to investigate our new models, it is a good test set since we can easily see the results of our method. We wish to identify the single Gaussian dimension and ignore the leptokurtotic dimensions. The leptokurtotic dimensions may be characterised as having long tails; if a residual can be created by removing the Gaussian direction from the data set, the residual will automatically be leptokurtotic. Thus we consider maximising the likelihood of the residual using the model

$$p(\mathbf{e}) = \frac{1}{Z}\exp(-|\mathbf{e}|^p) \quad \text{with } p < 2; \tag{26}$$

Then the cost function associated with this network is

$$J = -\log p(\mathbf{e}) = |\mathbf{e}|^p + K \tag{27}$$

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

$$\Delta W \propto -\frac{\partial J}{\partial W} = -\frac{\partial J}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial W} \approx \mathbf{y}(p|\mathbf{e}|^{p-1}\text{sign}(\mathbf{e}))^T \tag{28}$$

We have experimented with a number of values of p and report on simulations with $p = 1.5$. A typical result is shown in figure 1; the Gaussian direction is clearly identified.

An alternative perspective of the mechanics of the process may be given if we consider figure 2, which shows a Gaussian ($p = 2$) and a Laplacian ($p = 1$) distribution.

In the context of the above rules, we have

$$\Delta W \propto \mathbf{y}(p|\mathbf{e}|^{p-1}\text{sign}(\mathbf{e}))^T \tag{29}$$

Since the square root of the absolute value of the residuals is taken in this experiment ($p = 1.5$) we are giving greater emphasis to small residuals in the first part of the expression. This is where the Gaussian has more of its mass. The second term (sign($\mathbf{e}$)) acts equally on both and so the magnitude of the residuals does not affect this term i.e. the large tails found in the Laplacian distribution do not have the effect that they would have on a standard Hebbian rule or a rule with $p > 2$. Thus the Gaussian input in the above experiment is easily identified by this learning rule.

We have similar results with a data set containing 9 platykurtotic dimensions and one Gaussian dimension. We use the same learning rules as before but with a value of $p = 3$.

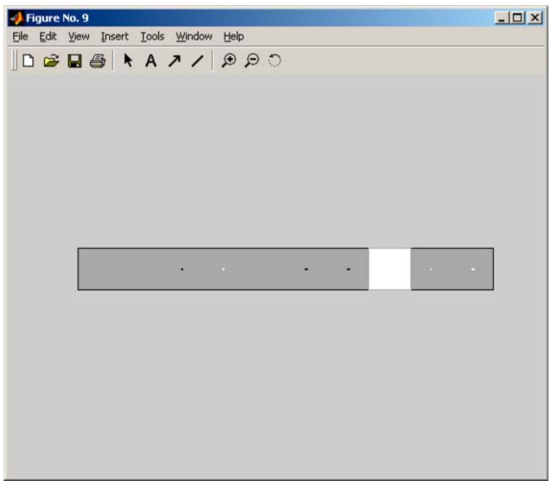


Figure 1. The Gaussian direction was the third among 9 leptokurtotic dimensions. It has clearly been identified in this Hinton map of the weights.

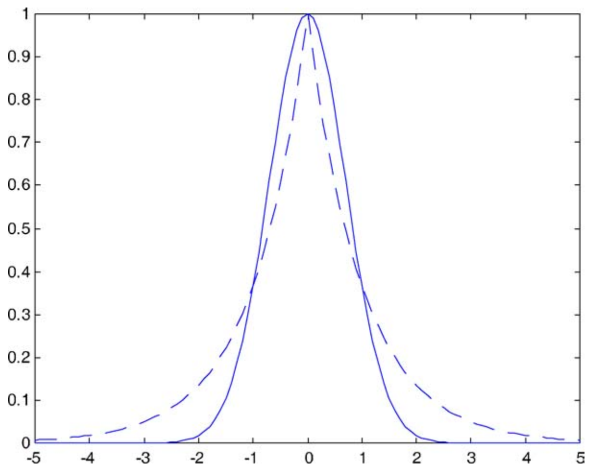


Figure 2. A Gaussian ($p = 2$) and a Laplacian ($p = 1$) distribution.

Artificial data set 2. This data set contains 9 independent Gaussian dimensions and one leptokurtotic dimension. This data set is more typical of data sets in which one might search for interestingness: most directions through this 10 dimensional data will be approximately Gaussian while there will be a single direction which is most positively kurtotic. Using the tactic which was successful on the first data set, we might suggest that maximising the likelihood of the residuals under a Gaussian model would be optimal. Our empirical finding, however, is that this criterion is not strong enough to identify the leptokurtotic dimension; we must use a model which is sub-Gaussian i.e. which maximises the likelihood of the residuals under a platykurtotic model. Now this is a wrong model but is more wrong for

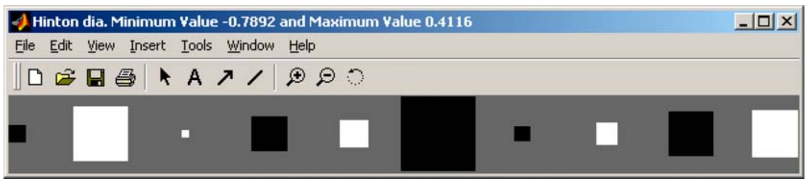

Hinton dia. Minimum Value -0.7892 and Maximum Value 0.4116
File Edit View Insert Tools Window Help

Figure 3. The fifth dimension from the right is the leptokurtotic dimension. It is always approximately identified but not as clearly as with the previous results.

the leptokurtotic dimension than for the Gaussian dimensions. It is necessary because of the fact that most projections of this data set are virtually Gaussian; it is very difficult to find the single leptokurtotic dimension in the high dimensional space. Over-penalising the positively kurtotic dimension is then somewhat effective. We may consider that, when using the model with $p = 3$, we are going to more readily penalise the leptokurtotic dimension since it has more samples in the tails of the distribution which are going to cause faster convergence to this dimension.

Typical results are shown in figure 3. The leptokurtotic dimension is fifth from the right and has been identified though far from clearly. We consider that this too is the effect of using the wrong model: some platykurtosis may be removed from the Gaussian dimensions; the Gaussian dimensions too may be thought of as less likely under this platykurtotic model.

The accuracy will be improved upon later using a form of anti-Hebbian learning.

Artificial data set 3. In this data set, we have 8 leptokurtotic dimensions and two Gaussians. Again we use a leptokurtotic model and attempt to identify the two Gaussian dimensions. Typical results are shown in figure 4 for which we used $p = 0.5$: the Gaussian dimensions are the fourth from the left and the third from the right. Typically the weights converge so that each is identifying the two dimensional subspace spanned by these two dimensions (as in the first and third parts of figure 4) though sometimes the two Gaussian dimensions are individually identified (as in the middle part of figure 4). We achieve 80% success in identifying the two Gaussian directions exclusively. A typical failure is shown in figure 5 , in which the first weight vector has converged to the subspace of the two Gaussian dimensions but in which the second weight vector is totally wrong.

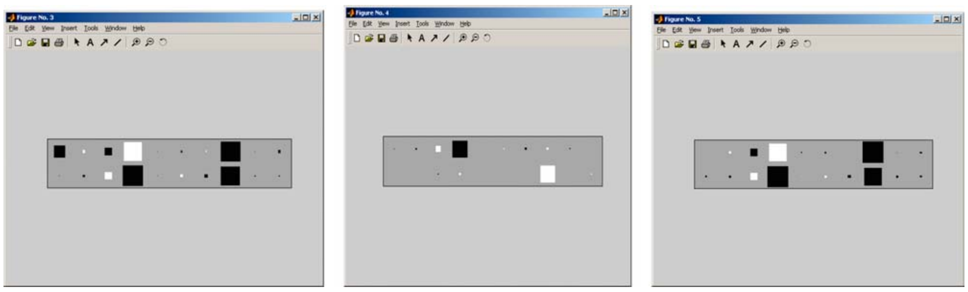


Figure 4. Identification of the two Gaussian directions.

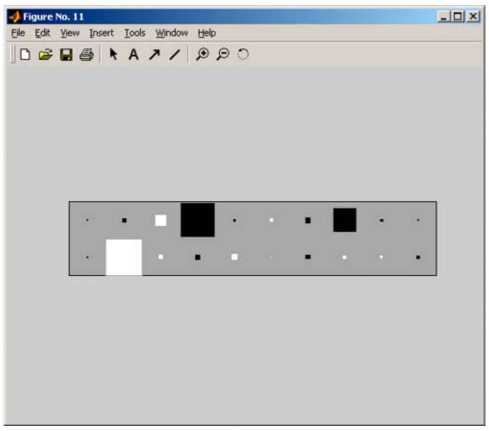


Figure 5. The first weight vector has identified the two leptokurtotic dimensions but the second is totally wrong $p = 0.5$.

The 80% success rate is not worthless for what is an exploratory data investigation method. Such methods are typically run over a data set several times starting from different initial conditions.

Minimum likelihood Hebbian learning

It is well known that the standard PCA rule: $\Delta W_{ij} = \eta y_i(x_j - \sum_k W_{kj} y_k)$ finds the first principal component (that with greatest eigenvalue) of a data set while $\Delta W_{ij} = -\eta y_i(x_j - \sum_k W_{kj} y_k)$ finds the first minor component (that with least eigenvalue) of a data set.

Therefore just as the Hebbian learning rule has an opposite known as the anti-Hebbian rule, we may change our rules so that

$$\Delta W \propto \frac{\partial J}{\partial W} = \frac{\partial J}{\partial \mathbf{e}} \frac{\partial \mathbf{e}}{\partial W} \approx -\mathbf{y}(p|\mathbf{e}|^{p-1}\text{sign}(\mathbf{e}))^T. \tag{30}$$

Now we may argue that, in doing so, we are aiming to minimise the likelihood of the residual given the current model. In detail, if the residual has probability density function

$$p(\mathbf{e}) = \frac{1}{Z}\exp(-|\mathbf{e}|^p). \tag{31}$$

and we denote the general cost function associated with this network as

$$J = E(-\log p(\mathbf{e})) = E(|\mathbf{e}|^p + K) \tag{32}$$

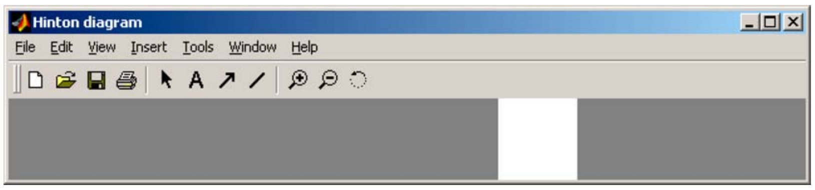


Figure 6. The platykurtotic dimension has been identified among the Gaussian dimensions.

where K is a constant, we may perform gradient ascent on J to get

$$\Delta W \propto \left.\frac{\partial J}{\partial W}\right|_{W(t-1)} = \left.\frac{\partial J}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial W}\right|_{W(t-1)} \approx E\big(-\mathbf{y}(p|\mathbf{e}|^{p-1}\text{sign}(\mathbf{e}))^T\big|_{W(t-1)}\big). \tag{33}$$

We are thus using our learning rules to make the residuals as unlikely as possible under the current model assumptions (determined by the p parameter). This is therefore particularly useful for data sets in which our previous results were less accurate. For example, when we have 9 Gaussian dimensions and 1 platykurtotic dimension (a data set on which we had rather inaccurate results previously) we get results as in figure 6 (with $p = 3$ in our minimum likelihood rule). By identifying and removing the platykurtotic dimension we are leaving a residual which has 0 kurtosis.

Note that with Minimum Likelihood Hebbian learning we are using the correct model for the distribution that we are seeking but minimising the probability of the residual being taken from this distribution. Thus we find and extract this distribution.

Other data sets

To show the power of the family of learning rules that we have derived we applied them to different data sets.

Bank data. To compare the new method of EPP with our previous neural implementation of the technique, we apply both methods on a small database of bank customers consisting of 1000 records each having 12 fields. Information held includes an unique identifier, age, sex, salary, type of area in which they live, whether married or not, number of children and then several fields of financial information such as type of bank account, whether they own a Personal Equity Plan etc. Figure 7 shows the projection of this data set on the filters found by the previous EPP network, Eqs. (5)–(8), when a cosine nonlinearity searching for clusters was used.

The network has clearly identified 4 clusters in the data set. Notice that neither of the one dimensional projections would have been sufficient to clearly identify four clusters (though one would have identified two clusters). Manual investigation of the clusters readily reveals that the clusters are forming on the place of residence field—each cluster is specific to one of RURAL, TOWN, INNER_CITY and URBAN sites. We should emphasise at this stage

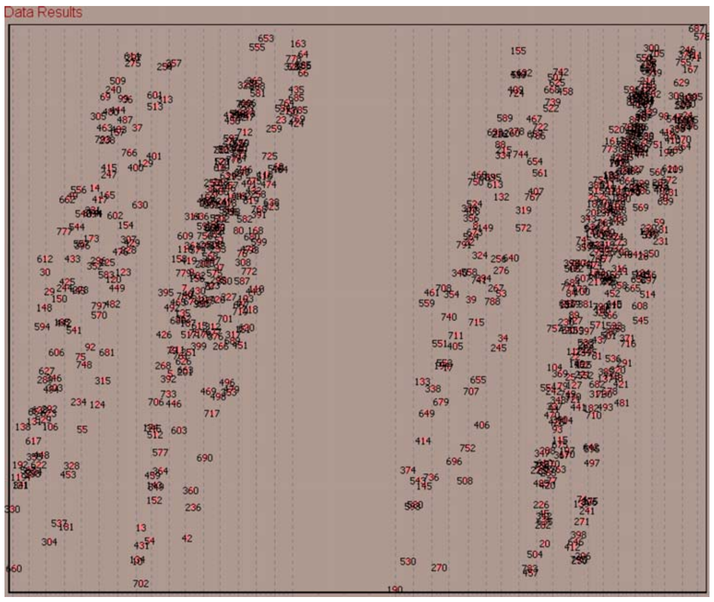


Figure 7. A two dimensional projection of the bank customer dataset clearly illustrating four clusters in the data set.

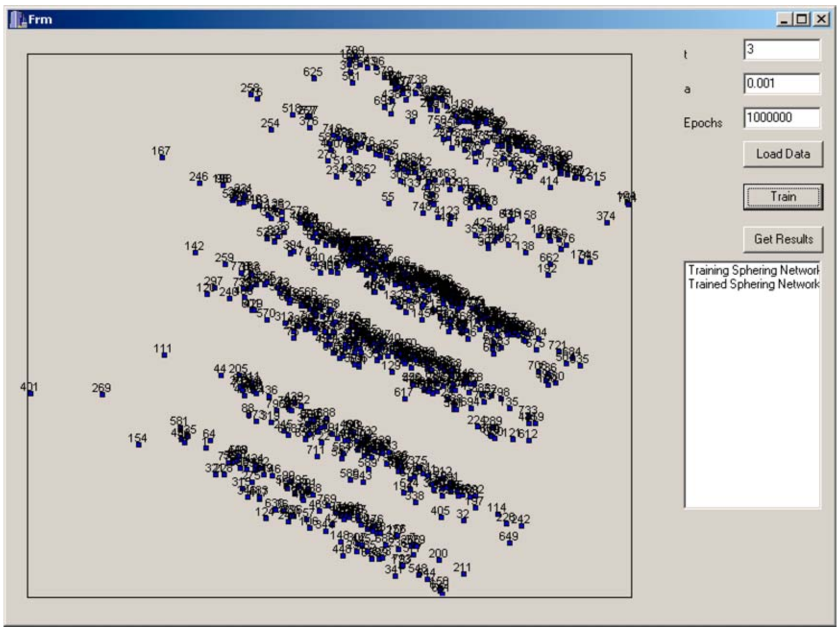
that this is far from the most interesting projection of the data set revealed by EPP but is being used for illustrative purposes.

Now we compare with the new EPP method; results are shown in figure 8.

We see far more structure in the projection from the Maximum Likelihood Hebbian rule than we had previously: not only are six main clusters found but also within each of these clusters we see distinct subgroups. This is a more interesting projection than that provide by the previous implementation of EPP which simple projects on AREA and SEX in the first axis and SALARY on the second. The ML method has separated each main cluster on AREA, CAR with the main sub-cluster of each diagonal separating on SEX. But we can see that the ML method has also many more well defined sub-clusters in its projections; these highlight groups of individuals whose profiles are slightly different from the main cluster. This shows the power of the ML rule as the number of sub-clusters identify far more texture in the projection than that identified by the previous implementation of EPP.

Astronomical data. The data consists of 65 colour spectra of 115 asteroids used by Howell et al. (1994). We have previously compared the performance of a variety of artificial neural networks on this data set (MacDonald et al., 1999).

The data set is composed of a mixture of the 52-colour survey by Bell et al. (1988) together with the 8-colour survey conducted by Zellner et al. (1985) providing a set of asteroid spectra spanning 0.3–2.5 μm. When this extended data set was compared by Howell et al. (1994) to the results in Tholen (1994) it was found that the additional refinement to the spectra lead to more classes in the taxonomy produced by Tholen. We have tested the networks



t 3
a 0.001
Epochs 1000000

Training Sphering Network
Trained Sphering Network

Figure 8. A two dimensional projection found by maximum likelihood Hebbian learning with $p = 0$.

on this data set looking at the differences in classification accuracy between clustering and projection networks. Standard PCA ($p = 2$) separates out the classes A and (some of) B but leaves most of the others in a single group (figure 9).

Maximum Likelihood Learning with $p < 2$ however shows a much greater separation of this central cluster (figure 10 was from a simulation with $p = 0.5$). If we compare figures 9 and 10, we see that both find the classes A and (some of) B easy to separate but Maximum Likelihood learning with $p < 2$ does spread the data out somewhat better.

Algae data. Our next data set is from a scientific study of various forms of algae some of which have been manually identified. Each sample is recorded as a 18 dimensional vector representing the magnitudes of various pigments. Some algae have been identified as belonging to specific classes which are numbered 1 to 9. Others are as yet unclassified and these are labelled 0. Figure 11 shows a projection of this data set onto the first two Principal Components. We can see that some separation of the classes has been achieved. However figure 12 shows a projection of the same data set onto the filters found using Minimum Likelihood Hebbian learning with $p = 1$; a rather better separation of the individual classes has been found.

Non-exponential distributions

In this paper, we have restricted our learning rules to those drawn from the exponential family of distributions. All of the artificial data sets above also came from this family of

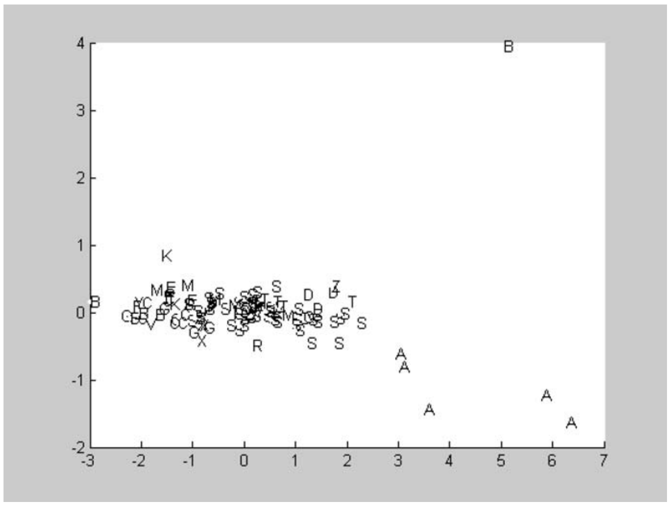


Figure 9. Projection of the asteroid data set onto the first two principal components.

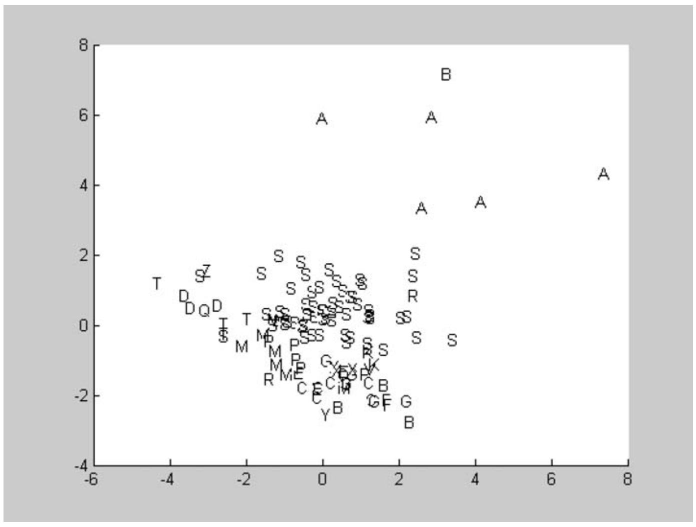


Figure 10. Projection of the asteroid data onto the filters found by maximum likelihood Hebbian learning with $p = 0.5$.

distributions and we might legitimately ask whether these rules will work on data sets which are not drawn from this family. To test whether the rules work on other data sets is totally an empirical question and is clearly dependent on the statistics of the data set in question. However to address this question in some small way we have repeated the above experiments with non exponential distributions. For example, Data set 4 was

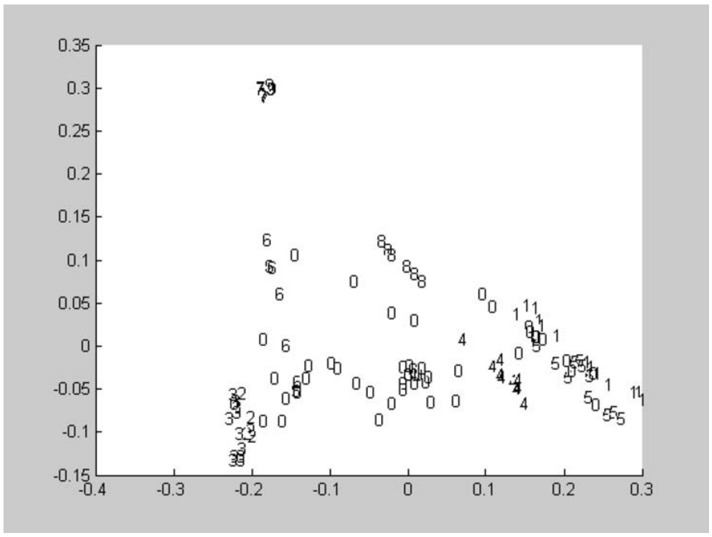


Figure 11. Projection of the algae data set onto the first two principal components.

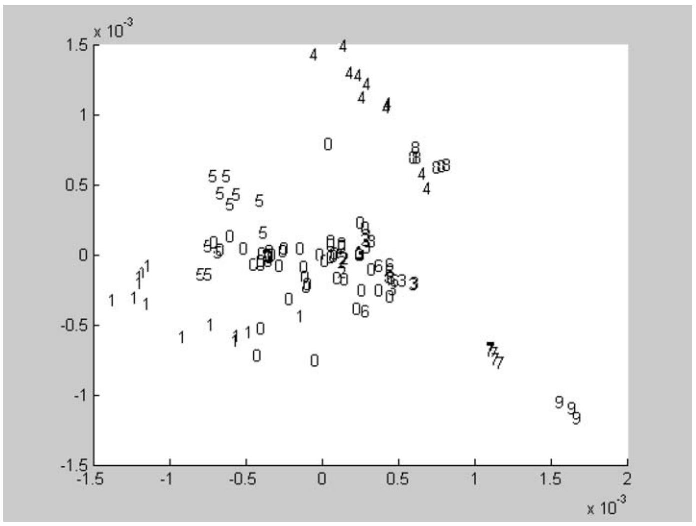


Figure 12. Projection of the algae data set onto the first two filters found using minimum likelihood Hebbian learning with $p = 1$.

slightly changed to 9 Gaussian dimensions and one drawn from the Beta (2, 2) distribution. We chose the Beta distribution since it is very malleable and we chose these parameters since it is then not unlike a Gaussian in shape (see figure 13).

Using $p = 3$ in our family of rules we consistently found the beta distribution. An example is shown in figure 14 in whch the beta distribution corresponds to the rightmost

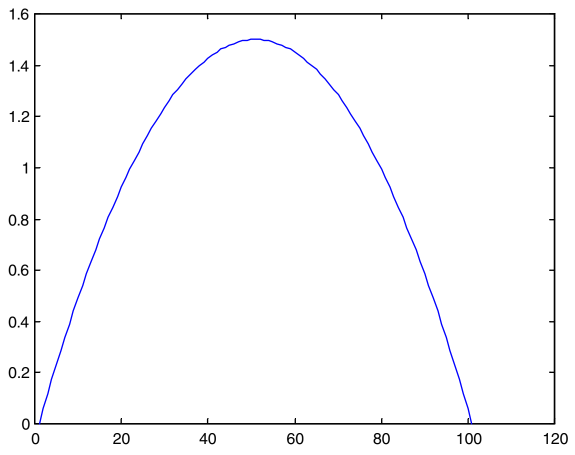


Figure 13. The Beta (2,2) probability density function.

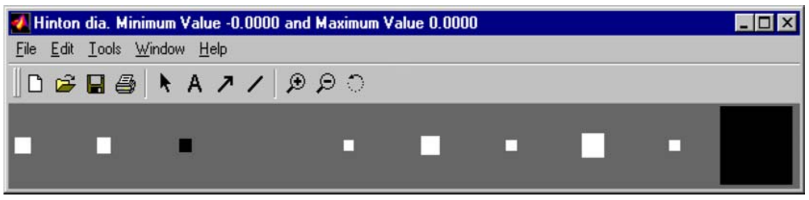

Hinton dia. Minimum Value -0.0000 and Maximum Value 0.0000
File Edit Tools Window Help

Figure 14. The dimension furthest to the right is drawn from the beta distribution. It is clearly identified.

dimension. The weights into this dimension are much greater than those into the other dimensions.

We might go on to ask whether the beta distribution has to be mixed with Gaussian distributions and so we create a similar data set with 9 platykurtotic exponential dimensions and one beta function dimension, ß (0.5, 0,5). We have used Minimum Likelihood Hebbian learning and $p = 3$. Since the β function has a non zero mean, this mean has been subtracted from the data. We used these values of the β parameters since the difference in kurtosis between the platykurtotic dimensions and the beta dimension is very small, $\sim$0.1 (see Table 1). A histogram of typical values drawn from this distribution is shown in figure 15. Note that the sphering of the data (which used PCA for this) caused the Beta distribution to be moved from third to last position (for the converged weights in figure 14) since it had smallest eigenvalue.

When sphered, the beta dimension is moved to last position since it has originally less variance than the exponential distributions.

In figure 16, we see that the dimension whose elements are drawn from the β distribution is clearly identified. Even more accurate results were obtained when using one β dimension with positive kurtosis and 9 leptokurtotic dimensions.

Table 1. The single β dimension (Dimension 3) has slightly more kurtosis than the others. Sometimes 3 is subtracted from the kurtosis values to give a Gaussian distribution kurtosis 0; this has not been done here.

Dimension	1	2	3	4	5	6	7	8	9	10
Kurtosis	1.3773	1.3872	**1.4947**	1.3792	1.3804	1.3734	1.3800	1.3904	1.3803	1.3787

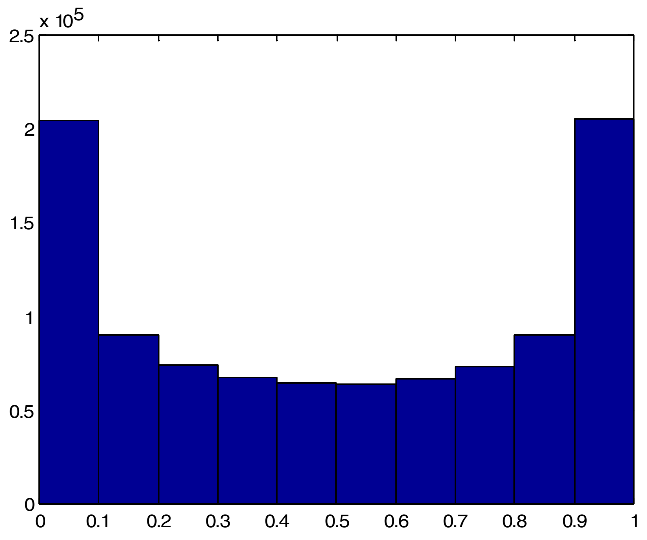


Figure 15. Histogram of the beta function for $w = 0.5$ and $v = 0.5$. This bimodal distribution is less negatively kurtotic than the platykurtotic exponential dimensions used in this experiment.

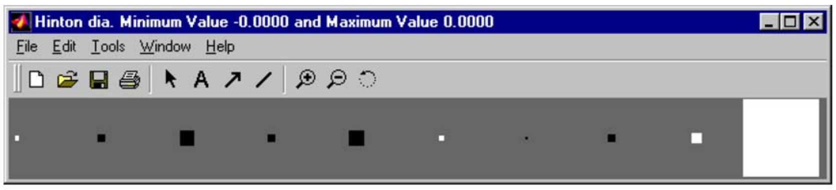


Figure 16. The first dimension from the right is the beta distribution. It has clearly been identified in this Hinton map of the weights.

We might also ask whether we need to use a member of the exponential family in order to derive learning rules appropriate for this data set. For example, we might ask if we can start with the probability density function of the Beta distribution and derive rules which will optimally find this distribution.

If the residual is draw from the Beta distribution, $B(v, \omega)$, with the following probability density function:

$$\mathbf{e}^{v-1}(1-\mathbf{e})^{\omega-1} = (\mathbf{x} - W\mathbf{y})^{v-1}(1 - \mathbf{x} + W\mathbf{y})^{\omega-1} \tag{34}$$

then if we wish to maximise the likelihood of the data with respect to the weights, we will perform gradient ascent using:

$$\frac{\partial p}{\partial W} = \mathbf{e}^{\upsilon-2}(1-\mathbf{e})^{\omega-2}\mathbf{y}(-(\upsilon-1)(1-\mathbf{e})+\mathbf{e}(\omega-1)) \tag{35}$$

which is rather a cumbersome rule. However for the case in which $\upsilon = \omega = 2$:

$$\frac{\partial p}{\partial W} = \mathbf{y}(-(1-\mathbf{e})+\mathbf{e}). \tag{36}$$

So the learning rule is simplified to:

$$\Delta W = \eta \mathbf{y}(2\mathbf{e}-1) \tag{37}$$

However experiments have shown that this rule is rather difficult to stabilise. We conjecture that the gradient of the logarithm of the probability density function (which we used for the exponential family) is somewhat more gradual than that of the pdf itself of the Beta distribution and that this is why the results with the platykurtotic members of the exponential family are better. Also it is worth stating that if we were to simply fit each distribution exactly with its analytically derived learning rule, this would give us little confidence in finding arbitrary empirical distributions in real data samples.

Comparing and mixing the two EPP methods

We now compare the effectiveness of these two algorithms on both artificial and real data sets. The artificial data is used to compare the speed of convergence of the algorithms in identifying interest in a data set since we know, in advance, exactly what sort of interesting structure is in the data set and can measure the progress of the algorithm towards identifying the structure. We will call the original algorithm the Higher Moments Algorithm.

Rate of convergence

In this section, we create a 10 dimensional data set in which 9 dimensions are drawn from a Gaussian distribution and one dimension from a uniform distribution. The uniform distribution is platykurtotic (has less kurtosis than the Gaussians) and so the higher moments algorithm can use $\mathbf{y}^3$ or more stably tanh(); the maximum likelihood method will use $p < 2$. The rate of convergence of the algorithms is shown in figure 17: the left figure shows the dot product of the weights with the ideal solution when the higher moments EPP algorithm with a tanh() nonlinearity is used while the right shows the convergence of the Maximum Likelihood EPP algorithm with $p = 1$. We see that the latter has extremely fast convergence but does not achieve an accuracy of more than 0.9 while the former, though it takes a little longer to get to the optimum, is much more accurate. This suggests that an algorithm which uses both rules might gain by having the best attributes of both and this is in fact the case.

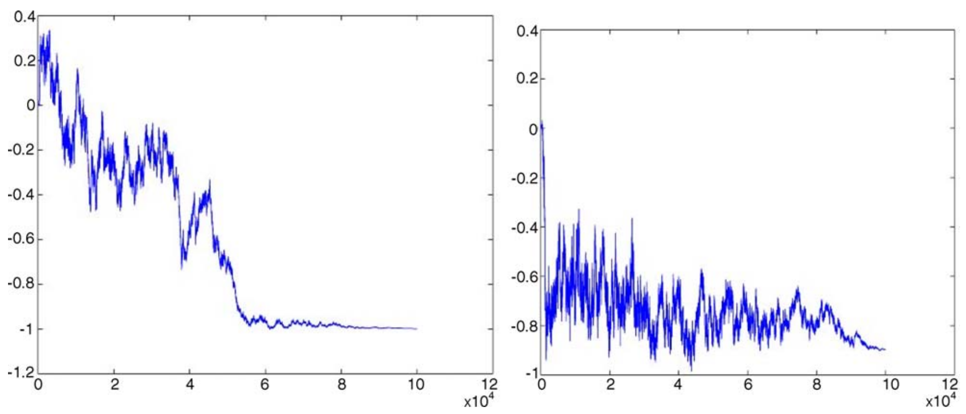


Figure 17. The left figure shows the convergence of the higher moments EPP algorithm while the right one shows the convergence of the maximum likelihood EPP algorithm in terms of the dot product to the ideal solution.

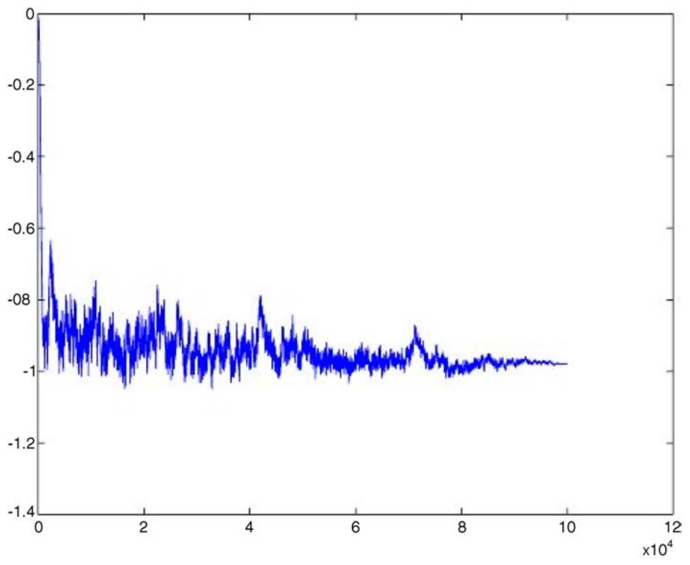


Figure 18. Convergence of the algorithm using the combined learning rule.

Figure 18 shows the convergence of an algorithm which uses a combination of these two rules i.e.

Feedforward:

$$y_i = \sum_{j=1}^{N} W_{ij} x_j, \ \forall_i$$

Feedback:

$$e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i$$

Weight change:

$$\Delta W_{ij} = \eta . f(y_i) \cdot \text{sign}(e_j)|e_j|^p$$

where $f()$ is the tanh () function in the experiment the results of which are shown in figure 18. We seem to be getting the best of both worlds with this combined method though it must be conceded that the combination is somewhat ad hoc. It is for this reason that we have not included results from this method elsewhere.

Conclusion

In this paper, we have derived a family of learning rules based on the probability density function of the residuals. This family of rules may be called Hebbian in that all use a simple multiplication of the output of the neural network with some function of the residuals after feedback. The power of the method comes from the choice of an appropriate function and what is appropriate is determined by the statistics of the data set. In particular, we showed how to choose a function to maximise the likelihood of the residuals under particular models of probability density functions. We now see that both the original PCA rule and the ε-insensitive rule (Lai et al., 2000) are merely particular cases of this class of rules. We have also shown that the rules are more akin to Exploratory Projection Pursuit and prefer to call them Maximum Likelihood Hebbian learning, believing that 'ε-insensitive PCA' does not do justice to the power of the method.

We have also shown how powerful Minimum Likelihood Hebbian learning is and indeed that this is, in some sense, even more closely related to EPP: the real power of these learning rules is in the context of exploratory data analysis. These are powerful new tools for the data mining community and should take their place along with existing exploratory methods.

Acknowledgments

The authors would like to acknowledge the comments of the two unknown reviewers whose insightful comments helped to improve this paper.

Note

1. This family was called an exponential family in Hyvärinen et al. (2002) though statisticians use this term for a somewhat different family.

References

Bell, J.F., Owensby, P.D., Hawke, B.R., and Gaffey, M.J. 1988. The 52 colour asteroid survey: Final results and interpretation. Lunar Planet Sci. Conf., XIX:57 (abstract).

Bishop, C.M. 1995. Neural Networks for Pattern Recognition. Oxford.

Diaconis, P. and Freedman, D. 1984. Asymptotics of graphical projections. The Annals of Statistics, 12(3):793–815.

Fyfe, C. 1993. PCA properties of interneurons. From Neurobiology to Real World Computing, Proceedings of International Conference on Artificial on Artificial Neural Networks, ICAAN 93, pp. 183–188.

Fyfe, C. and Baddeley, R. 1995. Non-linear data structure extraction using simple Hebbian networks. Biological Cybernetics, 72(6):533–541.

Fyfe, C. 1995. Negative feedback as an organising principle for artificial neural networks. Ph.D. Thesis, Strathclyde University.

Fyfe, C. and MacDonald, D. 2002. ε-insensitive Hebbian learning. Neurocomputing (Accepted for publication).

Howell, E.S., Merenyi, E., and Lebofsky, L.A., 1994. Using neural networks to classify asteroid spectra. Journal of Geophysical Research, 99(E5):10847–10865.

Hyvärinen, A. 2001. Complexity pursuit: Separating interesting components from time series. Neural Computation, 13:883–898.

Hyvärinen, A., Karhunen, J., and Oja, E. 2002. Independent Component Analysis. Wiley, ISBN 0-471-40540-X.

Karhunen, J. and Joutsensalo, J. 1994. Representation and separation of signals using non-linear PCA type learning. Neural Networks, 7:113–127.

Kashyap, R.L., Blaydon, C.C., and Fu, K.S. 1994. Stochastic approximation. In a Prelude to Neural Networks: Adaptive and Learning Systems, M. Jerry (Ed.), Mendel, Prentice Hall, ISBN 0-13-147448–0.

Lai, P.L., Charles, D., and Fyfe, C. 2000. Seeking independence using biologically inspired artificial neural networks. In Developments in Artificial Neural Network Theory: Independent Component Analysis and Blind Source Separation, M.A. Girolami (Ed.), Springer Verlag.

MacDonald, D., McGlinchey, S., Kawala, J., and Fyfe, C. 1999. Comparison of Kohonen, scale-invariant and GTM self-organising maps for interpretation of spectral data. European Symposium on Artificial Neural Networks (ESANN '99), 117–122.

Oja, E. 1989. Neural networks, principal components and subspaces. International Journal of Neural Systems, 1:61–68.

Oja, E., Ogawa, H., and Wangviwattana, J. 1992. Principal components analysis by homogeneous neural networks, Part 1. The Weighted Subspace Criterion. IEICE Transaction on Information and Systems, E75D:366–375.

Smola, A.J. and Scholkopf, B. 1998. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NeuroCOLT2 Technical Report Series.

Tholen, D. 1994. Asteroid taxonomy from cluster analysis of photometry. Ph.D. dissertation, University of Arizona.

Xu, L. 1993. Least mean square error reconstruction for self-organizing nets. Neural Networks, 6:627–648.

Zellner, B., Tholen, D.J., and Tedesco, E.F. 1985. The eight-colour asteroid survey: Results fro 589 minor planets. Icarus, 355–416.

Emilio 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 at University of Salamanca (Spain). His research interests center on artificial neural networks, with a particular focus on exploratory projection pursuit, self-organising maps and kernel methods. He is currently studying for a second Ph.D. at the Applied Computational Intelligence Research Unit at the University of Paisley, Scotland and is director of the Applied Computational Intelligence Research Group at the University of Burgos, Spain.

Donald MacDonald received his B.Sc. in Computer Science and Ph.D. from the University of Paisley in 1998 and 2002, respectively. He is currently a member of the Applied Computational Intelligence Research Unit at Paisley. His research interests include unsupervised learning and visualization of remote sensing images.

Colin Fyfe gained a B.Sc. (honours) in Mathematics from the University of Glasgow in 1971 and subsequently worked as a secondary school teacher for 17 years. He has since been awarded an M.Sc. in Information Technology (University of Strathclyde) in 1991, Master of Education (University of Glasgow) in 1992, Ph.D. (University of Strathclyde) in 1995. He is a member of the Academic Advisory Board for the International Computer Science Conventions group and has been Chair or Vice-Chair of several international conferences recently. He is on the Editorial Board of two International journals in computational intelligence and has had 9 Ph.Ds. completed in the last 4 years and is currently directng 8 others. He currently is Personal Professor at the University of Paisley and leads the Applied Computational Intelligence Research Unit. He is co-founder of 2020 Intelligence Ltd.