INITIALISING EXPLORATORY PROJECTION PURSUIT NETWORKS

César García-Osorio
Departamento de Ingeniería Civil
Universidad de Burgos,
Burgos, Spain
email: cgosorio@ubu.es

Colin Fyfe
Applied Computational Intelligence Research Unit
University of Paisley,
Scotland, United Kingdom
email: colin.fyfe@paisley.ac.uk

ABSTRACT
We have previously developed three artificial neural network methods of performing exploratory projection pursuit (EPP). EPP finds a low-dimensional linear projection of a high dimensional data set. A user can search for structure in the low-dimensional projection by eye. However, the projections found are typically very dependent on the initial conditions of the parameters (weights) in our artificial neural networks. In this study we initialise the EPP networks with values taken from Andrews’ Curves and show that the resulting networks converge very fast and reliably to the optimal projections.

KEY WORDS
Artificial neural networks, exploratory projection pursuit, data visualization, weights initialisation.

1 Introduction
Exploratory Projection Pursuit (EPP) is a technique for exploring high dimensional spaces. It can be thought of as an extension of Principal Component Analysis (PCA). PCA searches for filters in a data set which maximise the variance of the projections of the data onto those filters. It is a standard statistical technique which can be implemented using artificial neural networks [1, 2]. EPP can be thought of as an extension of PCA since it is a technique for finding projections which maximise some statistic over the data set. For example, if we wish to identify outliers in a data set, their presence might be revealed by a higher fourth moment (kurtosis) in the projections. EPP has also been implemented using artificial neural networks [3, 4].

2 EPP neural networks
We have over the last few years investigated a negative feedback implementation of PCA defined by the feedforward (1), the feedback (2) and the weight change (3):

\[ y_i = \sum_{j=1}^{N} w_{ij} x_j, \forall i \]  
\[ e_j = x_j - \sum_{i=1}^{M} w_{ij} y_i \]  
\[ \Delta w_{ij} = \eta e_j y_i \]

where \( w_{ij} \) is the weight linking the \( j \)th input to the \( i \)th output (there is \( N \) inputs and \( M \) outputs), and \( \eta \) is the learning rate, which will be annealed to zero over the course of training the network.

We have subsequently modified this network to perform clustering with topology preservation [6], to perform Factor Analysis [7, 8] and to perform Exploratory Projection Pursuit [3, 4].

2.1 The Output Function EPP network
Two common 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 basically measures if the distribution is symmetrical. 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 also have a negative kurtosis and therefore kurtosis can signal that a particular distribution shows evidence of clustering. Whilst these measures have their drawbacks as measures of deviation from normality (particularly their sensitivity to outliers), their simplicity makes them ideal for explanatory purposes.

In this paper, we investigate a method of biasing the learning in an EPP network so that the network is most likely to find directions which maximise the selected statistic. Essentially we do this by initialising the weights of the EPP network to specific values which make it most likely that the network will reach the global optimum rather than any local optima.
procedure. We have for \( N \) dimensional input data and \( M \) output neurons:

\[
y_i = \sum_{j=1}^{N} w_{ij} x_j
\]

(4)

\[
e_j = x_j - \sum_{i=1}^{M} w_{ij} y_i
\]

(5)

\[
r_i = f \left( \sum_{j=1}^{N} w_{ij} x_j \right) = f(y_i)
\]

(6)

\[
\Delta w_{ij} = \eta r_i e_j
\]

(7)

where \( r_i \) is the value of the function \( f() \) on the \( i \)th output neuron. Thus (7) may be written in matrix form as

\[
\Delta W(t) = \eta(t) [I - W(t) W^T(t)] x(t) f(x^T(t) W(t))
\]

(8)

where \( t \) is an index of time and \( I \) is the identity matrix.

In [9] is shown how (8) can be derived as an approximation to the maximisation of a function, \( J \), of the weights: \( J(W) = \sum_{i=1}^{M} E(\|x^T w_i\|_2) \) with \( E(.) \) the expectation operator.

### 2.2 The Maximum likelihood EPP network

Various researchers e.g. [10, 9] have shown that the learning rules (1)-(3) can be derived as an approximation to the best linear compression of the data. Thus we may start with the cost function

\[
J = 1^T E\{(x - Wy)^2\}
\]

(9)

which we minimise to get (3).

We may show that the minimisation of \( J \) is equivalent to minimising the negative log probabilities of the residual, \( e \), if \( e \) is Gaussian [11] and thus is equal to maximising the probabilities of the residual. Let \( p(e) = \frac{1}{2} \exp(-e^2) \).

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

\[
J = -\log p(e) = (e)^2 + K
\]

(10)

Intuitively, \( J \) gives a measure as to how probable the residual is under the current model parameters. Since the only parameters which can be changed are the weights which both feedforward and backward, these are the parameters which we must adapt in order to make the residuals more likely under the model. Therefore performing gradient descent on \( J \) we have

\[
\Delta W \propto -\frac{\delta J}{\delta W} = -\frac{\delta J}{\delta e} \frac{\delta e}{\delta W} \approx y(2e)^T
\]

(11)

where we have discarded a relatively unimportant term [9].

We have considered an extension of the above in [12, 13] with a more general cost function

\[
J = f_1(e) = f_1(x - Wy)
\]

(12)

Let us now consider the residual after the feedback to have probability density function

\[
p(e) = \frac{1}{Z} \exp(-|e|^p)
\]

(13)

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

\[
J = -\log p(e) = (e)^p + K
\]

(14)

where \( K \) is a constant, Therefore performing gradient descent on \( J \) we have:

\[
\Delta W \propto -\frac{\delta J}{\delta W} = -\frac{\delta J}{\delta e} \frac{\delta e}{\delta W} \approx y(|e|^{p-1} \text{sign}(e))^T
\]

(15)

We would expect that for leptokurtic residuals (more kurtotic than a Gaussian distribution), values of \( p < 2 \) would be appropriate, while platykurtic residuals (less kurtotic than a Gaussian), values of \( p > 2 \) would be appropriate. It is a common belief in the ICA community [14] that it is less important to get the 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 platykurtotic distribution.

Therefore the network operation is:

\[
y_i = \sum_{j=1}^{N} w_{ij} x_j, \forall i
\]

(16)

\[
e_j = x_j - \sum_{i=1}^{M} w_{ij} y_i
\]

(17)

\[
\Delta w_{ij} = \eta y_i \text{sign}(e_j)|e_j|^p
\]

(18)

Now the nature and quantification of the interestingness is in terms of how likely the residuals are under a particular model of the probability density function of the residual. As with standard EPP, we also sphere the data before applying the learning method to the sphered data.

### 2.3 The Combined EPP network

The two EPP algorithms were derived from different perspectives and each used changes to a Principal Component Analysis neural network in different parts of the weight change algorithm. This might suggest that we can get the best of both worlds by combining these algorithms. In [5] we investigated the convergence of an algorithm which uses the same feedforward and feedback as in the previous networks, but which has the following weight change

\[
\Delta w_{ij} = \eta f(y_i) \text{sign}(e_j)|e_j|^p
\]

(19)

where

- \( f(y) = \tanh(y) \) and \( p = 1 \) for the bimodal data.
- \( f(y) = y - \tanh(y) \) and \( p = 3 \) for the leptokurtotic data.
3 Andrews Curves

Andrews [15] described his curves in 1972, early on in the computing era; it an interesting observation that he thought it necessary to counsel “an output device with relatively high precision ... is required”. Current standard PC software is quite sufficient for the purposes. The method is another way to attempt to visualise and hence to find structure in high dimensional data. Each data point \( \mathbf{x} = \{x_1, x_2, ..., x_n\} \) defines a function

\[
 f_x(t) = x_1/\sqrt{2} + x_2 \cos(t) + x_3 \sin(t) + \ldots 
\]

and this function is then plotted for \( -\pi < t < \pi \). Thus each data point may be viewed as a line between \( -\pi \) and \( \pi \). If there is structure in the data, it may be visible in the Andrews’ Curves of the data. An example of Andrews’ Curves on the well known iris data set is shown in Figure 1. The data set is four dimensional (only the first four terms of (20) are used). We see that one type of iris is easily identifiable from the other two but there is some difficulty in differentiating between these two. Thus \( t = 3 \) gives us a value for a linear projection of the data which differentiates one type of data. But the fact that we cannot see any clear distinction between the other two types of iris in the figure does not necessarily mean that it is not possible to find a projection which will do so: the Andrews parameters are limited by the properties of the trigonometric functions (e.g. \( \sin^2() + \cos^2() = 1 \) means that we have a very constrained relationship between the parameters of the projection for \( x_2 \) and \( x_3 \)). Thus our conjecture is that we can identify some structure in the Andrews plots and use this as an initial condition for an EPP network.

The EPP networks that we use get the most accurate results when the spherings are applied to the data. On the contrary, the Andrews’ Curves obtained using the sphered data lack the necessary structure that make them useful as a visualization tool. That does not happen when the raw data is used. So in the experiments, that we present bellow, the Andrews’ curve over the original data are used. Then the vector obtained after selecting an interesting \( t \) value is transformed in accordance to the process of spherings before being used to initialize the network weights.

4 Initialising EPP Networks with Andrews Curves

To study the improvements that could be obtained using the Andrews’ curves we performed many experiments using artificial data sets as in [3]. In four experiments we used 10 dimensional inputs, 9 independent Gaussian dimensions and one platykurtotic dimension obtained from a Gaussian distribution and then randomly adding and subtracting the same quantity. This gives a bimodal distribution with two clusters. Following that procedure 300 points were obtained and then they were sphered. In the fifth experiment we used 19 gaussians instead of 9, obtaining in this way a more complex data set. Each experiment comprised 20 simulations in which the networks were trained using 20000 samples and initialized using the weights obtained from the Andrews’ curves, and 20 more simulations with random initial weights (between zero and one). To present the results the mean of the 20 simulations is used.

In all cases but one (see below) the value of the initial learning rate was 0.01 which was annealed to zero during the course of the simulation. These artificial data sets are convenient since we know what direction we are searching for (e.g. \( (1,0,0,0, ...0) \)) and it is very easy to check convergence.

In the first experiment, we used the Output Function EPP algorithm [3] with the function \( f(y) = \tanh(y) \) in the learning rule. Figure 2 shows the good convergence obtained when the Andrews’ curves are used to initialize the network weights. With random weights the convergence is very poor i.e. often non-existent.

Now the \( \tanh() \) function was used as a stable approximation to \( -y^3 \). We may now report that with a second set of experiments using \( -y^3 \) and a learning rate of 0.0001, we achieved 100% convergence when using the Andrews’ Curves initialisation (Figure 3) and 100% NaN errors when not.

The aim of the third experiment was to investigate the effect of the initialization in the Maximum Likelihood EPP algorithm. We can see in Figures 4 and 5 that with the initialization of the weights, little is gained. Indeed, this EPP algorithm actually moves the weights away from the optimal values so that what was taken, in previous experiments, to be lack of accuracy on the part of the algorithm is actually due to the algorithm learning something other than that which it was designed to learn. We see that whether we initialise with the Andrews’ Curves values or randomly, the weights converge to the same values.

The fourth experiment makes use of the learning rule which is a combination of the Output Function and the
Maximum Likelihood learning rules. As we expected the influence of the initialization was null, since this learning rule exhibits a very good convergence even with random initial weights.

The last experiment uses a more complex dataset than the previous ones. On this occasion we tried to identify a bimodal distribution immersed in 19 gaussian distributions. Again the advantages of the initialization can be appreciated. Figure 6 shows the results with the initialization from the Andrews’ Curves, in the Figure 7 the 8 best results without the initialization are shown.

5 Conclusion

We have, in this paper, investigated the improvements that can be achieved using initial weights other than random weights. The method that we used to do the initialisation is based on Andrews’ Curves that enables the user to choose a promising projection where the EPP network can start its search.

We have shown how with appropriate initialisation weights an algorithm with very bad convergence can achieve a very good convergence. The initialisation even enables us to use algorithm that without it do not exhibit convergence. Interestingly, the results of Figures 4 and 5 suggest that the results which we had previously ascribed to poor convergence actually are due to the algorithm responding to statistics other than that which it is designed for. This aspect has thrown new light on the convergence of our EPP algorithms and is worthy of future theoretical investigation.

References

Figure 2. The mean of 20 simulations that use the Output Function EPP Algorithm \( f(y) = \tanh(y) \) using weight initialization from specific Andrews’ Curves values. We also show one standard deviation either side of the mean.

Figure 3. The mean of 20 simulations that use the Output Function EPP Algorithm \( f(y) = -y^3 \) using weight initialization from specific Andrews’ Curves values. We also show one standard deviation either side of the mean.

Figure 4. The convergence of the 20 simulations that use the Maximum Likelihood EPP Algorithm using the weight initialization from the Andrews’ Curves.

Figure 5. The convergence of the 20 simulations that use the Maximum Likelihood EPP Algorithm initialising with random weights.

Figure 6. Convergence of the Output Function algorithm with \( f(y) = \tanh(y) \) using 19 gaussians distributions and one bimodal distributions using the Andrews weight initialization. We also show one standard deviation either side of the mean.

Figure 7. Convergence of the Output Function algorithm with \( f(y) = \tanh(y) \) using 19 gaussians distributions and one bimodal distributions. The mean of the best 8 experiments (and one standard deviation either side) when the weights are randomly initialised.