Dynamic Selection of Model Parameters in Principal Components Analysis Neural Networks

Ezequiel López-Rubio, Juan Miguel Ortiz-de-Lazcano-Lobato, María del Carmen Vargas-González and **José Miguel López-Rubio**[1]

Abstract. One of the best known techniques for multidimensional data analysis is the Principal Components Analysis (PCA). A number of *local PCA* neural models have been proposed to partition an input distribution into meaningful clusters. Each neuron of these models uses a certain number of basis vectors to represent the principal directions of a particular cluster. Most of these neural networks are unable to learn the number of basis vectors, which is specified *a priori* as a fixed parameter. This leads to poor adaptation to input data. Here we develop a method where the number of basis vectors of each neuron is learned. Then we apply this method to a well known local PCA neural model. Finally, experimental results are presented where the original and modified versions of the neural model are compared.

1 INTRODUCTION

The Principal Components Analysis is a multispectral data analysis technique, which is aimed to obtain the principal directions of the data, i. e., the maximum variance directions (see [6], [8]). Hence, if we have a D-dimensional input space, the PCA computes the K principal directions, where $K<D$. This allows important dimensionality reductions by selecting $K<<D$. It has been proved that PCA is an optimal linear technique for dimensionality reduction, in the mean sense (see [6]).

The original method, sometimes called *Karhunen-Loève (KL) transform* or *global PCA*, considers the entire input distribution as a whole. A number of *local PCA methods* have been proposed to partition the distribution into meaningful clusters (see [7], [11], [14]). These methods have been widely used in the context of transform coding to compress multispectral and multilayer images (see [12], [15]). Furthermore, they are used in visualization of high-dimensional data, which requires mapping to a lower dimension (typically, $K\leq 3$). Nonlinear extensions of PCA [2] have also been used for this task. Nevertheless, local PCA procedures do not address the problem of selecting the correct number of basis vectors K. This problem has been studied in the context of global PCA by several authors (for example, [1] and [9]). The lack of a clear criterion to select K leaves this value as a parameter to be adjusted *a priori* by the human operator. Our goal here is to propose a solution for this issue, when a local PCA neural model is used.

This paper is organized as follows. Section 2 presents a method to estimate the number of basis vectors. The application of this method to a local PCA neural model is developed in Section 3.

[1] School of Computer Engineering, University of Málaga. 29071 Málaga. Spain. email: ezeqlr@lcc.uma.es

Section 4 is devoted to the proofs of some important properties of the method. We make a short discussion of the advantages of our proposal in Section 5. Sections 6 and 7 are devoted to experimental results and conclusions, respectively.

2 THE EXPLAINED VARIANCE METHOD

The PCA method uses the *covariance matrix* to analyze the input distribution. The covariance matrix of an input vector $\mathbf{x}$ is defined as

$$\mathbf{R} = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x} - E[\mathbf{x}])^T] \quad (1)$$

where $E[\cdot]$ is the mathematical expectation operator and we suppose that all the components of $\mathbf{x}$ are real random variables. In addition, the mean vector $\mathbf{e}=E[\mathbf{x}]$ is also used. PCA methods form a vector base $B=\{\mathbf{b}_h \mid h=1,...,K\}$ with the eigenvectors $\mathbf{b}_h$ corresponding to the K largest eigenvalues of $\mathbf{R}$, where K is a parameter which is specified before the start of the learning process. This vector base encomprises the K principal directions of the input distribution corresponding to $\mathbf{R}$. In the case of local PCA neural models, each neuron j has a covariance matrix $\mathbf{R}_j$ and a mean vector $\mathbf{e}_j$.

The method considers a variable number of basis vectors $K_j(t)$, which is computed independently for each neuron j. This number reflects the intrinsic dimensionality of the data in the receptive field of neuron j (i. e., the set of inputs for which the neuron j is the winner). There are several algorithms to approximate the intrinsic dimensionality of the data ([5]; [16]). Most of them are based in the analysis of the eigenvalues $\lambda_j^p(t)$ of the covariance matrix $\mathbf{R}_j(t)$, $p=1,\ldots,D$.

Each neuron of a local PCA network represents a cluster of data, where $\mathbf{e}_j$ and $\mathbf{R}_j$ are estimations of the mean and the covariance matrix of the cluster, respectively. If the cluster of data were represented only by its sample mean, i.e., $K=0$, then the mean squared error corresponding to this representation would be:

$$MSE_0 = E\left[\left\|\mathbf{e}_j - \mathbf{x}\right\|^2\right] = E\left[\sum_{q=1}^{D}\left(e_{jq} - x_q\right)^2\right] \quad (2)$$

Let $\mathbf{b}_i$ be the ith principal direction of neuron j, with $\|\mathbf{b}_i\|=1$ (the index j is dropped for simplicity). If we use a vector base $B_Z=\{\mathbf{b}_1,\ldots,\mathbf{b}_Z\}$, where Z is the number of basis vectors, the mean squared error associated with this representation is given by:

$$MSE_Z = E[\left\|\mathbf{x}-\mathbf{e}_j - Orth\left(\mathbf{x}-\mathbf{e}_j, B_Z\right)\right\|^2] \quad (3)$$

where *Orth* means orthogonal projection:

$$Orth\left(\mathbf{v}, B_z\right) = \sum_{h=1}^{Z}\left(\mathbf{b}_h{}^T\mathbf{v}\right)\mathbf{b}_h \quad (4)$$

Note that if Z=0 we get the maximum possible mean squared error. The goal here is to select a number of basis vectors K which ensures that at least a fraction α of the maximum mean squared error is removed,

$$K_j = \min\{Z \in \{0,1,...,D\} \mid MSE_0 - MSE_Z \geq \alpha\, MSE_0\} \quad (5)$$

where $\alpha \in [0,1]$. Next we show how this can be achieved. First we rewrite (3) in terms of projections on the principal directions:

$$MSE_Z = E\left[\left\|\left(\sum_{i=1}^{D}\left(\mathbf{b}_i^T\left(\mathbf{x}-\mathbf{e}_j\right)\right)\mathbf{b}_i\right) - \left(\sum_{i=1}^{Z}\left(\mathbf{b}_i^T\left(\mathbf{x}-\mathbf{e}_j\right)\right)\mathbf{b}_i\right)\right\|^2\right] =$$

$$E\left[\left\|\sum_{i=Z+1}^{D}\left(\mathbf{b}_i^T\left(\mathbf{x}-\mathbf{e}_j\right)\right)\mathbf{b}_i\right\|^2\right] \quad (6)$$

Since the principal directions are orthonormal, this means that

$$MSE_Z = E\left[\sum_{i=Z+1}^{D}\left(\mathbf{b}_i^T\left(\mathbf{x}-\mathbf{e}_j\right)\right)^2\right] \quad (7)$$

From (7) and the definition of variance,

$$MSE_Z = \sum_{i=Z+1}^{D}\mathrm{var}\left(\mathbf{b}_i^T\left(\mathbf{x}-\mathbf{e}_j\right)\right) \quad (8)$$

On the other hand, we also know from PCA that the variance of the projection of the input minus its mean onto the *i*th principal direction $\mathbf{b}_i$ equals the *p*th largest eigenvalue of $\mathbf{R}$:

$$\forall p \in \{1,2,...,D\}, \quad \mathrm{var}\left(\mathbf{b}_i^T\left(\mathbf{x}-\mathbf{e}_j\right)\right) = \lambda_j^p \quad (9)$$

Substitution of (9) into (8) yields

$$MSE_Z = \sum_{p=Z+1}^{D}\lambda_j^p \quad (10)$$

If we remember that e_{jp}=E[x_p], equation (2) can be rewritten in terms of variances:

$$MSE_0 = \sum_{p=1}^{D}\mathrm{var}\left(x_p\right) \quad (11)$$

From PCA we know that the sum of variances equals the trace of $\mathbf{R}$:

$$MSE_0 = trace\left(\mathbf{R}_j\right) = \sum_{p=1}^{D}\lambda_j^p \quad (12)$$

Finally, we substitute (10) and (12) into (5):

$$K_j = \min\left\{Z \in \{0,1,...,D\} \mid \sum_{p=1}^{D}\lambda_j^p - \sum_{p=Z+1}^{D}\lambda_j^p \geq \alpha\, trace\left(\mathbf{R}_j\right)\right\} \quad (13)$$

This can be simplified as follows:

$$K_j = \min\left\{Z \in \{0,1,...,D\} \mid \sum_{p=1}^{Z}\lambda_j^p \geq \alpha\, trace\left(\mathbf{R}_j\right)\right\} \quad (14)$$

Note that there is no need to compute all the eigenvalues of $\mathbf{R}_j$, but only the K_j largest ones. If we add the time instant *t* we get the equation to be used in practice:

$$K_j(t) = \min\left\{Z \in \{0,1,...,D\} \mid \sum_{p=1}^{Z}\lambda_j^p(t) \geq \alpha\, trace\left(\mathbf{R}_j(t)\right)\right\} \quad (15)$$

The quotient between $\lambda_j^p(t)$ and the trace of $\mathbf{R}_j(t)$ is the amount of variance explained by the *p*th principal direction of $\mathbf{R}_j(t)$. So, we can see the parameter α as the amount of variance which we want the neurons to explain. Hence, we select $K_j(t)$ so that the amount of variance explained by the directions associated to the $K_j(t)$ largest eigenvalues is at least α.

3 APPLICATION TO LOCAL PCA

In this section we apply the explained covariance method to the local PCA model by Kambathla and Leen [7]. This model uses a fixed number *M* of neurons. At each time instant *t*, the neuron *j* stores a mean vector $\mathbf{e}_j(t)$ and a covariance matrix $\mathbf{R}_j(t)$. The values corresponding to the next time instant $\mathbf{e}_j(t+1)$ and $\mathbf{R}_j(t+1)$ are obtained as follows. First of all, the input distribution is partitioned into *M* regions C_j (*j*=1,2,…,*M*):

$$C_j = \{\ \mathbf{x} \mid \forall i = 1,...,M,\ d_j(\mathbf{x}) \leq d_i(\mathbf{x})\ \} \quad (16)$$

where $d_i(\mathbf{x})$ is the reconstruction distance (i.e., the projection error) for neuron *i*,

$$d_i(\mathbf{x}) = \left\|\mathbf{x}-\mathbf{e}_i(t) - Orth\left(\mathbf{x}-\mathbf{e}_i(t), B^i(t)\right)\right\|^2 \quad (17)$$

and $B^i(t)$ is the vector base associated with the principal directions of $\mathbf{R}_j(t)$. Then the mean vectors and the covariance matrices are recomputed with respect to the regions C_j:

$$\mathbf{e}_j(t+1) = \frac{1}{n_j}\sum_{p=1}^{n_j}\mathbf{x}_p \quad (18)$$

$$\mathbf{R}_j(t+1) = \frac{1}{n_j}\sum_{p=1}^{n_j}\left(\mathbf{x}_p - \mathbf{e}_j(t+1)\right)\left(\mathbf{x}_p - \mathbf{e}_j(t+1)\right)^T \quad (19)$$

where n_j is the number of input samples in C_j.

The application of the explained variance method consists in the addition of a final step where the number of basis vectors $K_j(t+1)$ is computed for each neuron j by using equation (15). That is, the vector base $B^j(t+1)$ will have $K_j(t+1)$ basis vectors. Hence the size of the vector bases is controlled by the explained variance method, while in the original approach by Kambathla and Leen the number of basis vectors is fixed and it is the same for all the neurons.

The algorithm can be stated as follows:

1. Obtain initial estimations $\mathbf{e}_j(0)$ and $\mathbf{R}_j(0)$ for all the neurons j. This is achieved by computing the mean vectors and covariance matrices of clusters of data chosen randomly.
2. Use (15) to compute the initial numbers of basis vectors $K_j(0)$.
3. Partition the input distribution by using (16).
4. Compute $\mathbf{e}_j(t+1)$ and $\mathbf{R}_j(t+1)$ with (18) and (19), respectively.
5. Use (15) to compute the new numbers of basis vectors $K_j(t+1)$.
6. If convergence has been reached, or the error is below some threshold, stop the algorithm. Otherwise, go to step 3.

4 PROPERTIES

Next we prove some important properties of the presented method. First of all, two bounds of the projection errors are considered. In particular, the bound obtained in Proposition 2 allows to perform a fast check of the mean squared error without the need to compute the error for all the input samples. The third result expresses how the number of basis vectors K_j is controlled by the parameter α. Finally, it is shown that the model developed here reduces to the k-means algorithm ([3], [4]) when $\alpha=0$.

Proposition 1 *For every time instant t and every neuron j, the sum of squared errors of the input samples assigned to neuron j is bounded as follows:*

$$\sum_{p\in C_j}\left(d_j(\mathbf{x}_p)\right)^2 \le (1-\alpha)\sum_{p\in C_j}\left\|\mathbf{x}_p-\mathbf{e}_j\right\|^2 \tag{20}$$

Proof: From equation (2) and the definition of mean we know that

$$n_j\ MSE_0 = \sum_{p\in C_j}\left\|\mathbf{x}_p-\mathbf{e}_j\right\|^2 \tag{21}$$

n_j is the number of input samples in C_j. Also, from equation (3) and the definition of mean,

$$n_j\ MSE_{K_j} = \sum_{p\in C_j}\left(d_j(\mathbf{x}_p)\right)^2 \tag{22}$$

We recall from equation (5) that

$$MSE_0 - MSE_{K_j} \ge \alpha\ MSE_0 \tag{23}$$

Then we rearrange the inequality:

$$MSE_{K_j} \le (1-\alpha)\ MSE_0 \tag{24}$$

Next we multiply both sides by n_j:

$$n_j\ MSE_{K_j} \le (1-\alpha)\ n_j\ MSE_0 \tag{25}$$

By substitution of equations (21) and (22) into (25) we get (20), as required.

Proposition 2 *For every time instant t, the sum of squared errors of the samples of the input distribution C is bounded as follows:*

$$\sum_{p\in C}\left(d(\mathbf{x}_p)\right)^2 \le (1-\alpha)\left(\sum_{p\in C}\left\|\mathbf{e}-\mathbf{x}_p\right\|^2 - \sum_{j=1}^{M} n_j\left\|\mathbf{e}-\mathbf{e}_j\right\|^2\right) \tag{26}$$

where $\mathbf{e}$ *is the overall mean of the input distribution.*

Proof: First we expand the summation of the squared errors by considering each neuron separately:

$$\sum_{p\in C}\left(d(\mathbf{x}_p)\right)^2 = \sum_{j=1}^{M}\sum_{p\in C_j}\left(d_j(\mathbf{x}_p)\right)^2 \tag{27}$$

On the other hand, we can sum equation (20) for all neurons j to get

$$\sum_{j=1}^{M}\sum_{p\in C_j}\left(d_j(\mathbf{x}_p)\right)^2 \le (1-\alpha)\sum_{j=1}^{M}\sum_{p\in C_j}\left\|\mathbf{x}_p-\mathbf{e}_j\right\|^2 \tag{28}$$

It is well known ([13]) that:

$$\sum_{p\in C}\left\|\mathbf{e}-\mathbf{x}_p\right\|^2 = \sum_{j=1}^{M} n_j\left\|\mathbf{e}-\mathbf{e}_j\right\|^2 + \sum_{j=1}^{M}\sum_{p\in C_j}\left\|\mathbf{x}_p-\mathbf{e}_j\right\|^2 \tag{29}$$

Then we substitute (29) into (28):

$$\sum_{j=1}^{M}\sum_{p\in C_j}\left(d_j(\mathbf{x}_p)\right)^2 \le (1-\alpha)\left(\sum_{p\in C}\left\|\mathbf{e}-\mathbf{x}_p\right\|^2 - \sum_{j=1}^{M} n_j\left\|\mathbf{e}-\mathbf{e}_j\right\|^2\right) \tag{30}$$

Finally, we substitute (27) into (30) to get (26), as desired.

Proposition 3 *Let $z\in\{0,1,\ldots,D\}$. If $\alpha\in[0,z/D]$, then for every neuron j and every time instant t, it holds that $0\le K_j(t)\le z$.*

Proof: As the eigenvalues are sorted in decreasing order, it holds that:

$$\forall p\in\{1,\ldots,z\}\forall q\in\{z+1,\ldots,D\},\ \lambda_j^p(t)\ge\lambda_j^q(t) \tag{31}$$

$$\Rightarrow (D-z)\sum_{p=1}^{z}\lambda_j^p(t) \ge z\sum_{q=z+1}^{D}\lambda_j^q(t) \tag{32}$$

Next we add $z\sum_{p=1}^{z}\lambda_j^p(t)$ to both sides:

$$D\sum_{p=1}^{z}\lambda_j^p(t) \geq z\sum_{q=1}^{D}\lambda_j^q(t) \tag{33}$$

$$\sum_{p=1}^{z}\lambda_j^p(t) \geq \frac{z}{D}\sum_{q=1}^{D}\lambda_j^q(t) \tag{34}$$

By the hypoteses, we have that $\alpha \in [0,z/D]$, so:

$$\sum_{p=1}^{z}\lambda_j^p(t) \geq \alpha\sum_{q=1}^{D}\lambda_j^q(t) \tag{35}$$

Then by (12), (15) and (35) we have:

$$0 \leq K_j(t) \leq z \tag{36}$$

Corollary 1 *If we take $\alpha=0$, the local PCA model with the explained variance method reduces to the k-means algorithm.*

Proof: We use $z=0$ in the previous proposition, and we get $\alpha=0 \Rightarrow K_j(t)=0$ for every neuron j. Then the covariance matrices are not used in the competitive phase, so it reduces to that of k-means. The estimated mean vectors $\mathbf{e}_j(t)$ play the role of the k-means.

5 DISCUSSION

The proposed model has the following advantages:

a) It learns the number of basis vectors which are needed to represent the input distribution with a specified accuracy. That is, the basis vectors are selectively removed or added to the neurons as needed.

b) Proposition 2 can be used to perform a fast check of the mean squared error. It is not necessary to compute the projection error of the input samples, but only to compute the differences between the mean vectors of the different neurons $\mathbf{e}_j$ and the overall mean $\mathbf{e}$ (see equation 26). It must be noted that the term $\sum_{p\in C}\|\mathbf{e}-\mathbf{x}_p\|^2$ can be precomputed prior to the initialization of the neural network, so there is no computational load associated to it. This way to bound the mean squared error would be useful when it must be ensured that the error falls below some fixed quantity. For example, in a multispectral image compression application we would control the termination of the algorithm (step 6) with this technique, if a prespecified quality is required in the compressed image.

c) The learned number of basis vectors can be used to obtain intrinsic dimensionality information. Low values of K_j mean that the region represented by neuron j has a low dimensionality, and vice versa.

d) It is a generalization of the k-means algorithm, as proven in the previous section.

6 EXPERIMENTAL RESULTS

We have designed a set of experiments to test the dimensionality reduction performance of our method. This unsupervised task has been used to compare our proposal with the original approach by Kambhatla and Leen [7] and the Probabilistic Principal Components Analysis (PPCA) neural model by Tipping and Bishop [14]. The PPCA networks consider a fixed number of basis vectors K for all neurons, like Kambathla and Leen. Both are aimed to reduce the projection error, as our approach. Hence we use the mean normalized squared projection error (MNSE) to measure the performance of the three systems with independence from the scale of the input vectors:

$$MNSE = \frac{1}{N}\sum_{p=1}^{N}\frac{(d_j(\mathbf{x}_p))^2}{\|\mathbf{x}_p\|^2} \tag{37}$$

In order to reduce the dimensionality of the input distribution, the projection error minimization must be achieved with the minimum possible number of basis vectors. So, we are interested in the relation between these two values.

Our experiments have used a freely-accessible data set from the NASA Earth Observatory [10]. This data set is made up of climatic images taken every month by satellites. We have selected 6 months (from January 1988 to June 1988), and 14 parameters for each month. So we have 14 images per month, each with 360x180 pixels. These 14 images are combined to form a single multisensor image, where every pixel is a vector with 14 components. The values of the components are real numbers in the interval [0,1]. Each of the vectors is an input sample. Every experiment starts by presenting the input samples of a month to a network. When the training is finished, the projection error is computed for all the input samples of the month. Then these projection errors are used to compute the mean error of the month. Finally, the mean errors from the 6 months are averaged to yield the final mean error.

Table 1. Average CPU time used to train the original local PCA network (fixed *K*), in seconds.

# neurons	K=1	K=2	K=3	K=4	K=6	K=8
2	6070	7762	8822	7928	9063	9325
4	7520	8358	8695	10172	11625	13166
8	10347	12304	14686	16336	19058	21875
16	18343	22056	26096	33357	35891	41261

Table 2. Average CPU time used to train the proposed local PCA approach (dynamic *K*), in seconds.

# neurons	α=0.5	α=0.6	α=0.7	α=0.8	α=0.9
2	6783	8371	7956	8974	9336
4	8835	8725	9381	10911	11937
8	13429	13464	14224	15698	16575
16	21466	22547	23639	25939	29081

The parameters of the local PCA network have been selected as follows. We have run 20 iterations of the method, i.e., we have presented the data of the considered month 20 times to the network. The original approach needs the number of basis vectors K to be specified, so we have carried out experiments with K=1, 2, 3, 4, 6 and 8. All these values of K have been used with 2, 4, 8 and 16 neurons. Our approach has been tested with α=0.5, 0.6, 0.7, 0.8 and 0.9, and the same numbers of neurons.

For the PPCA model, we have run 10 iterations of the method, i.e., the data of the considered month has been presented 10 times to the network. We have selected this reduced number of iterations because further processing showed no improvement. The values of K and the numbers of neurons have been the same as in the original local PCA approach.

The plots of the error MNSE versus the number of basis vectors K is shown in Figs. 1 to 4. The number of basis vectors K has been averaged over all the neurons (this is the reason because fractional values appear). Please note that nearer to the coordinate origin is better. For our approach ('dynamic K'), the results corresponding

to lower α appear to the left, because a smaller amout of variance can be explained with fewer basis vectors. We can see that our dynamic approach has better performance than the original local PCA with fixed *K* and the PPCA in all the tests.

Table 3. Average CPU time used to train the PPCA networks, in seconds.

# neurons	K=1	K=2	K=3	K=4	K=6	K=8
2	967	1007	1051	1049	1074	1092
4	1844	1928	2090	2106	2088	2119
8	3458	3636	3852	3940	3993	4079
16	8270	8394	9084	9142	9420	9122

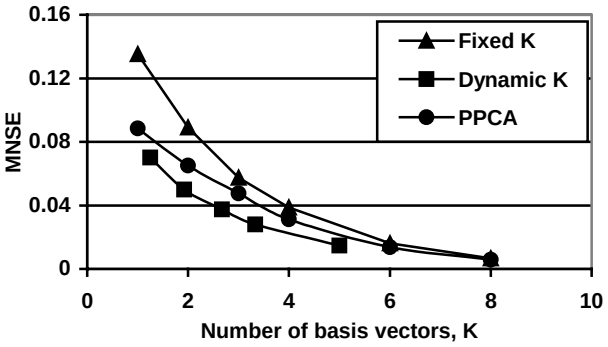

Figure 1. Network performance results with 2 neurons.

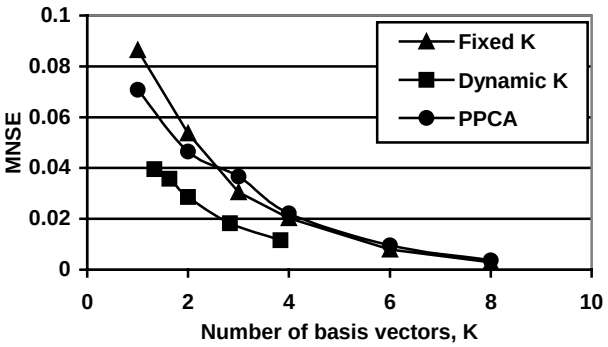

Figure 2. Network performance results with 4 neurons.

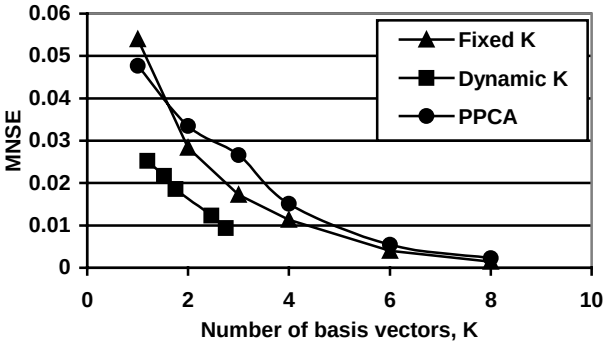

Figure 3. Network performance results with 8 neurons.

The CPU time required by the three approaches can be seen on Tables 1 to 3. Our proposal is slightly faster than the original PCA, due to the removal of unnecesary basis vectors, while PPCA is the fastest (but only at the expense of a very reduced performance, as seen before).

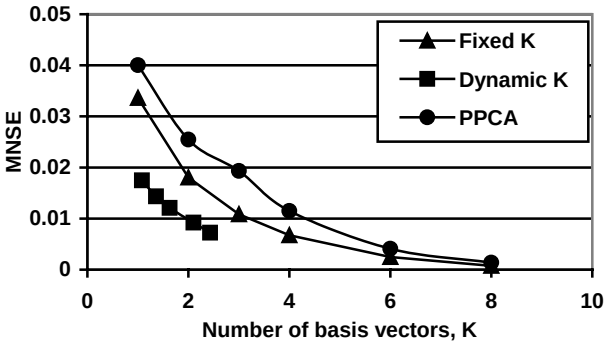

Figure 4. Network performance results with 16 neurons.

7 CONCLUSIONS

We have presented the explained variance method to select the number of basis vectors in local PCA neural networks. With this method, every neuron is able to modify its behaviour to adapt to the local dimensionality of the input distribution, leading to greater plasticity. This learning is controlled by a parameter which specifies the desired amount of explained variance. Some important properties of this method have been proved and discussed, including bounds of the mean squared errors. Finally, experimental results have been presented, which show that our approach outperforms two well-known models with a fixed number of basis vectors.

REFERENCES

[1] P. Besse, 'PCA stability and choice of dimensionality', *Statistics & Probability Letters*, **13** (5), 405–410, (1992).

[2] C.M. Bishop, M. Svenson and C.K.I. William, 'GTM: The generative topographic mapping', *Neural Computation*, **10**(1), 215-234, (1998).

[3] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, 1973.

[4] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, CA, 1990.

[5] K. Fukunaga and D.R. Olsen, 'An Algorithm for Finding Intrinsic Dimensionality of Data', *IEEE Trans. Computers*, **20** (2), 176–183, (1971).

[6] I.T. Jolliffe, *Principal Component Analysis*, Springer-Verlag, Berlin, 1986.

[7] N. Kambhatla and T.K. Leen, 'Dimension Reduction by Local Principal Component Analysis', *Neural Computation*, **9** (7), 1493–1516, (1997).

[8] M. Kendall, *Multivariate Analysis*, Charles Griffin&Co, London, 1975.

[9] T.P. Minka, 'Automatic choice of dimensionality for PCA', *Advances in Neural Information Processing Systems*, **13**, 598–604, (2001).

[10] NASA Earth Observatory, *Data & Images* [online]. Available: *http://earthobservatory.nasa.gov/Observatory/datasets.html* (November 28, 2002).

[11] S. Roweis and Z. Ghahramani, 'A Unifying Review of Linear Gaussian Models', *Neural Computation*, **11**, 305–345, (1999).

[12] J.A. Saghri, A.G. Tescher and J.T. Reagan, 'Practical Transform Coding of Multispectral Imaginery', *IEEE Signal Processing Magazine*, 32–43, (January 1995).

[13] H. Spath, *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*, Wiley, New York, 1980.

[14] M.E. Tipping and C.M. Bishop, 'Mixtures of Probabilistic Principal Components Analyzers', *Neural Computation*, **11**, 443–482, (1999).

[15] D. Tretter and C.A. Bouman, 'Optimal Transforms for Multispectral and Multilayer Image Coding', *IEEE Trans. Image Processing*, **4** (3), 296–308, (1995).

[16] P.J. Verveer and R.P.W. Duin, 'An Evaluation of Intrinsic Dimensionality Estimators', *IEEE Trans. Pattern Analysis and Machine Intelligence*, **17** (1),.81–86, (1995).