10 Oriented Principal Component Analysis for Feature Extraction

Abstract- Principal Components Analysis (PCA) is a popular unsupervised learning technique that is often used in pattern recognition for feature extraction. Some variants have been proposed for supervising PCA to include class information, which are usually based on heuristics. A more principled approach is presented in this paper. Suppose that we have a pattern recogniser composed of a linear network for feature extraction and a classifier. Instead of designing each module separately, we perform global training where both learning systems *cooperate* in order to find those components (Oriented PCs) useful for classification. Experimental results, using artificial and real data, show the potential of the proposed learning system for classification. Since the linear neural network finds the optimal directions to better discriminate classes, the global-trained pattern recogniser needs less Oriented PCs (OPCs) than PCs so the classifier's complexity (e.g. capacity) is lower. In this way, the global system benefits from a better generalisation performance.

Index Terms- Oriented Principal Components Analysis, Principal Components Neural Networks, Co-operative Learning, Learning to Learn Algorithms, Feature Extraction, Online gradient descent, Pattern Recognition, Compression.

Abbreviations- PCA- Principal Components Analysis, OPC- Oriented Principal Components, PCNN- Principal Components Neural Networks.

1. Introduction

In classification, observations belong to different classes. Based on some prior knowledge about the problem and on the training set, a pattern recogniser is constructed for assigning future observations to one of the existing classes. The typical method of building this system consists in dividing it into two main blocks that are usually trained separately: the feature extractor and the classifier.

Traditionally the feature extractor, which aim to reduce the complexity of the original problem by means of projecting the input patterns to a lower-dimensional space, was completely hand-crafted since it is rather specific to the problem. The main problem with this ad-hoc approach is that classification accuracy largely depends on how well the manual feature selection is performed. The advent of cheap computers, large databases, and new powerful learning machines has changed this way of thinking over the last decade. Automatic feature extraction is nowadays employed in many difficult real-world problems like handwriting digit recognition (Choe et al., 1996) (Sirosh, 1995) (Sckölkopf et al., 1998).

Principal Components Analysis (PCA) (Jolliffe, 1986) is a powerful technique of multivariate analysis for dimensionality reduction in an automatic fashion. It has a wide variety of different applications including cluster analysis, visualisation of high-dimensional data, regression and feature extraction in pattern recognition. The most simple way of using PCA as a feature extractor is to replace the original patterns (of dimension p) by their first m ($m<p$) high-variance PCs in the input of the classifier. These PCs are estimated using a training set by solving an eigenvalue problem (chapter 11; Press et al., 1992) or by using adaptive algorithms like Principal Component Neural Networks (PCNN) (Diamantaras & Kung, 1996).

The principal problem of PCA in classification is that there is no reason to believe that the separation between classes will be in the direction of the high-variance PCs for any classification problem. The first few PCs will only be useful in those cases where the intra- and inter- class variations have the same dominant directions or inter-class variations are clearly larger than intra-class variations. Otherwise, PCA will lead to a partial (or even complete) loss of discriminatory information. This is also the problem of any other unsupervised technique involved in feature extraction since they take no account of class label in the target data.

Several simple variants of PCA that use class information have been proposed. The most popular group of techniques for pattern recognition based on PCA is subspace classifiers (Wold, 1976) (Fukunaga & Koontz, 1970) (Oja, 1983). They compute PCs separately for each class and describe them by a low-dimensional principal component subspace. The main drawback of these

methods is that they do not retain information about the relative differences between classes since populations with similar PCs lead to very poor classification accuracy. Attempts to overcome this limitation have appeared by introducing coupling terms between classes but they mainly rely on heuristics.

A more principled approach is presented in this paper. Instead of the usual separate training of the feature extractor and the classifier, we propose a global and iterative training of both systems. In this framework, the classifier regularises the principal component extractor in order to achieve a better separation of classes in the feature space. Hence, the supervised and unsupervised learning systems co-operate in order to find a global solution to the classification problem. The classifier repeatedly learns to form class boundaries in a feature space that is progressively transformed according to the instructions (e.g. misclassification error) given by the classifier itself in order to better separate classes.

In the next section, we will introduce the basics of our approach to the oriented PCA for classification. To achieve a transformation from the original space to a feature space useful for classification, oriented principal component analysis (OPCA) is presented as the solution of a minimization problem of a loss function composed by the usual average reconstruction error plus an additional cost function that depends on the misclassification error of a classifier that works in the transformed (feature) space. Section 3 presents a constructive neural network that extracts multiple OPCs in a sequential manner. In Section 4, the global learning algorithm of a pattern recognizer is presented. This learning system is composed of the oriented principal component neural network of the last section connected in cascade with any classifier whose outputs are differentiable respect their inputs. In section 5, the classifier that we use in conjunction with the oriented principal component neural network in the simulations is introduced. Section 6 shows experimental results using artificial and real databases. Finally, some discussion and conclusion are given in sections 7 and 8.

2. Basics of OPCA for Classification

PCA performs a linear transformation **U** from an input space X of dimension p to a feature space Z of dimension m (m<p). This projection is defined by

$$\mathbf{z} = \mathbf{U}^{\mathrm{T}}\mathbf{x}, \qquad \mathrm{U} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_{\mathrm{m}} \end{bmatrix} \tag{1}$$

where T denotes transpose, subject to the constraint

$$\left\|\mathbf{u}_j\right\| = 1, \quad \mathbf{u}_j{}^T\mathbf{u}_i = 0, \quad j = 1...m, \quad i \neq j \tag{2}$$

Given a set of samples $\{\mathbf{x}_i, i = 0,..., N-1\}, \mathbf{x}_i \in \Re^p$ with the mean of $\mathbf{x}$ to be unknown, $\mathbf{U}$ is found by solving the following least squares problem:

$$\min_U L_{PCA}(\mathbf{U}) = \min_U \frac{1}{N}\sum_{i=0}^{N-1}\left\|\left(I - \mathbf{U}\mathbf{U}^T\right)\left(\mathbf{x}_i - \langle\mathbf{u}\rangle\right)\right\|^2 \tag{3}$$

where $\mathbf{I}$ is the identity matrix and

$$\langle\mathbf{x}\rangle = \frac{1}{N}\sum_{i=0}^{N-1}\mathbf{x}_i \tag{4}$$

The optimal solution $\mathbf{U}^*$ is any linear combination of the first m eigenvectors with the largest eigenvalues (or highest variance) of the sample covariance matrix,

$$\hat{\mathbf{R}}_{XX} = \frac{1}{N-1}\sum_{i=0}^{N-1}\left(\mathbf{x}_i - \langle\mathbf{x}\rangle\right)\left(\mathbf{x}_i - \langle\mathbf{x}\rangle\right)^T \tag{5}$$

PCA as a feature extraction technique is only useful for those classification problems in which high-variance PCs coincide with the direction of separation between classes. Otherwise, low-variance PCs can retain important discriminatory information. To better obtain a linear transformation that takes into account not only those directions of maximum variance but also the directions of maximum class separation, it is necessary to modify the original optimisation problem by incorporating class information.

We turn, for a moment, to the classification part of the problem and assume that the point $\mathbf{x}$ is then a pattern to recognise that belongs to one of the C classes. To classify it, we first transform $\mathbf{x}$ using the linear transformation $\mathbf{U}$, that originally is composed of the first m PCs, and then pass $\mathbf{z}=\mathbf{U}^T\mathbf{x}$ through a maximum classifier Ψ. This classifier assigns $\mathbf{z} \in \Re^m$ to a class label from a finite set $\Im=\{1, ..., C\}$ according to the following rule:

$$\text{assign } \mathbf{z} \text{ to class label } j \Leftrightarrow d_i(\mathbf{z}) = \max_{j=1,\ldots,C} d_j(\mathbf{z}) \tag{6}$$

where d_j j=1,...,C are the so-called discriminant functions which are normalised to sum the unity ($\sum_{j=1}^{C} d_j(\mathbf{z}) = 1$). (For instance, in a MLP the values of these discriminant functions are simply the activation levels of the neurons of the output layer.)

Suppose then we have N observations pairs $T=\{(\mathbf{x}_i, \mathbf{y}_i), \quad i = 0,\ldots, N-1\}$ where $\mathbf{x_i} \in \Re^p$ is a random sample taken from the input space X and belongs to one of the classes and $\mathbf{y_i}$ is an indicator variable. If $\mathbf{x_i}$ belongs to the nth class, the nth coefficient of $\mathbf{y_i}$ is equal to 1 and the other coefficients are equal to 0. We can define as a measure of performance for classification to be minimised, the usual empirical Bayes risk defined by

$$B_{emp}(\Psi) = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=1}^{C} 1(\Psi(\mathbf{z}_i) = j)\sum_{k=1}^{C} R_{kj}\hat{P}(y_{ki} = 1 | Z = \mathbf{z}_i) \tag{7}$$

where $\mathbf{z}_i = B\mathbf{x}_i$, R_{kj} is the risk of classifying X as class j when the true class is k, **1**(condition) is 1 if the condition is true and 0 otherwise. We assume that there are equal risks for misclassifications, i.e.

$$R_{kj} = \begin{cases} 1 & \text{if } k \neq j \\ 0 & \text{if } k = j \end{cases} \tag{8}$$

Then the empirical Bayes risk classifier becomes an empirical maximum posterior classifier since

$$B_{emp}(\Psi) = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=1}^{C} 1(\Psi(\mathbf{z}_i) = j)\left(1 - \hat{P}(y_{ji} = 1 | Z = \mathbf{z}_i)\right) = \frac{1}{N}\sum_{i=0}^{N-1}\left(1 - \max_{j=1,\ldots,C}\hat{P}(y_{ji} = 1 | Z = \mathbf{z}_i)\right) \tag{9}$$

Its optimal value minimises the average number of misclassifications,

$$\Psi_{\text{empirical Bayes}} = \arg\min_{\Psi} \frac{1}{N}\sum_{i=0}^{N-1} 1(\Psi(\mathbf{z}_i) \neq \text{class label}(\mathbf{x}_i)) \tag{10}$$

Hence we can use as a measure of performance for classifier design functions like the cross-entropy error for multiple classes (Section 6.9, Bishop, 1995)

$$L_{cross-entropy}(\Psi) = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{l=1}^{C} y_{li} \ln d_l(\mathbf{z}_i) \tag{11}$$

since its absolute minimum occurs when all the training points are correctly classified, that is when $d_l(\mathbf{z_i})=y_{li}$ for all l and i.

As we pointed out before, our goal here is to obtain a linear transformation of the input space based on principal component analysis but also useful for classification. The simplest way of modifying the original PCA to incorporate class information is by adding a penalty term $L_{classifier}(\Psi, \mathbf{U})$ in the original cost function in the following way:

$$L_{OPCA}(\mathbf{U}) = L_{PCA}(\mathbf{U}) + \lambda L_{classifier}(\Psi, \mathbf{U}) \tag{12}$$

Here $L_{PCA}(\mathbf{U})$ is the cost function defined in (3), $L_{classifier}(\Psi, \mathbf{U})$ denotes any function of the misclassification error of a classifier Ψ that works in the feature space $\mathbf{Z}=\mathbf{U}^T\mathbf{X}$ and the parameter λ controls the degree to which the penalty term influences the solution. The resulting projections from minimising $L_{OPCA}(\mathbf{U})$ are a compromise between achieving PCs and those projections useful for discriminating in the feature space. If λ is small, the projections will be close to PCs. As λ grows, they will be oriented from PCs to those projections that imrpoves the classication accuracy of the classifier that discriminates in the feature space.

In order to provide correct information about how the projections are useful to classification, we must re-train the classifier Ψ that works in the feature space, each time we modify the linear transformation $\mathbf{U}$. In this way, the regularisation process to obtain the oriented PCs (OPCs) might form part of a global training process in which the feature extractor and the classifier are simultaneously trained in a co-operative manner.

We might start obtaining a pattern recogniser based on PCA. Then, by means of an iterative procedure, PCs would be progressively oriented to those directions in which the transformed training points can be better discriminated. The term $L_{classifier}(\Psi, \mathbf{U})$ added to the cost function of the feature extractor $L_{OPCA}(\mathbf{U})$ might provide the sufficient feedback information from the classifier's output to the feature extractor in order to compute the OPCs. This could be accomplished since the classifier Ψ would be re-trained for each new computed linear

transformation **U**. A mathematical description of this co-operative learning algorithm is shown below.

Basic Co-operative Learning Algorithm for OPCA plus Classification

Given a training set $T=\{(\mathbf{x}_i, \mathbf{y}_i), \quad i=0,...,N-1\}$

Step 1. k=0

Step 2. Design of the initial recogniser that uses standard PCA

Step 2.1. Compute $\mathbf{U}_*^0$ in order to minimise

$$L_{PCA}(\mathbf{U}^0)=\frac{1}{N}\sum_{i=1}^{N}\left\|\left(I-\mathbf{U}^0\mathbf{U}^{0T}\right)\left(\mathbf{x}_i-\langle\mathbf{x}\rangle\right)\right\|^2 \tag{13}$$

(given the initial conditions: $\mathbf{U}^0$ is a random matrix)

Step 2.2. Compute Ψ_*^0 in order to minimise

$$L_{classifier}(\Psi^0,\mathbf{U}_*^0)=\frac{1}{N}\sum_{i=0}^{N-1}1\left(\Psi^0\left(\mathbf{U}_*^{0T}\mathbf{x}_i\right)\neq y_i\right) \tag{14}$$

Step 3. Iterative Design of the Recogniser that uses Oriented PCA

Step 3.1. Compute $\mathbf{U}_*^{k+1}$ in order to minimise

$$L_{OPCA}(\mathbf{U}^{k+1})=L_{PCA}(\mathbf{U}^{k+1})+\boldsymbol{l}\,L_{classifier}(\Psi_*^{k},\mathbf{U}^{k+1}) \tag{15}$$

(given the initial conditions: **U** is $\mathbf{U}_*^k$)

Step 3.2. Compute Ψ_*^{k+1} in order to minimise

$$L_{classifier}(\Psi^{k+1},\mathbf{U}_*^{k+1})=\frac{1}{N}\sum_{i=0}^{N-1}1\left(\Psi^{k+1}\left(\mathbf{U}_*^{k+1T}\mathbf{x}_i\right)\neq y_i\right) \tag{16}$$

(given the initial conditions: Ψ is Ψ_*^k)

Step 3.3. k=k+1

Step 3.4. **If** *stopping conditions* are met, **then** go to Step 4 **else** go to Step 3.1.

Step 4. End

3. A Constructive Neural Network for OPCA

Given a classifier Ψ_*^k that employs a linear transformation $\mathbf{U}_*^k$ (pxm), our problem is to compute a new matrix $\mathbf{U}_*^{k+1}$ that minimises equation (15). Suppose we can compute $\partial L_{\text{classifier}}(\Psi, \mathbf{U})/\partial \mathbf{U}$. Then the most direct approach is to derive a learning PCNN based on a constrained gradient descent.

Since the optimisation process based on gradient descent evolves in a time-discrete scale ($n \geq 0$) , the linear transformation $\mathbf{U}^{k+1}[n]$ modifies the situation of transformed points that the fixed classifier Ψ_*^k must discriminate. Since this classifier was built using $\mathbf{U}_*^k$ and this matrix is the initial value of $\mathbf{U}^{k+1}[n]$ in the learning equations, it is convenient to impose some restrictions in the way of computing the OPCs. It makes necessary that the sequence $\{\mathbf{U}^{k+1}[n],\ n \geq 0\}$ smoothly varies since the efficacy of $\partial L_{\text{classifier}}(\Psi_*^k(\bullet), \mathbf{U}^{k+1}[n])/\partial \mathbf{U}^{k+1}[n]$ seems restricted to the range of values of $\mathbf{U}^{k+1}[n] \approx \mathbf{U}_*^k$.

A simple way of limiting the degree of change in $\mathbf{U}^{k+1}[n]$ as n grows is to estimate OPCs in a sequential manner. This constructive computation will make the feature points $\mathbf{z}=\mathbf{U}^{k+1}[n]\mathbf{x}$ to be moved only in one dimension each time k, so the classifier Ψ_* will be able to better discriminate according to the old location than if all components of $^{k+1}[n]$ vary at the same

Another desirable constraint would be that the PCNN always estimates the same components for a given initial condition $\mathbf{U}^{k+1}[0] = \mathbf{U}_*^k$, classifier Ψ_*^k and value of λ. However, this can only happens when λ=0 we force to the PCNN to exactly estimate PCs instead of a linear combination of PCs. This behaviour will make possible to increase the convergence rate of the learning algorithm since if the learning system exactly computes PCs when λ=0 then it is expected that $\mathbf{U}_*^{k+1} \approx \mathbf{U}_*^k$.

There are two kinds of problems for extracting multiple PCs using linear neural networks (p.281, Kung, 1993): 1) estimate the linear subspace spanned by the m PCs of highest variance, 2) estimate exactly these first m PCs. Many different architectures and learning algorithms have been proposed having this goal in mind (Diamantaras & Kung, 1996). In our case, the learning equations must be directly derived from equation (12) and multiple PCs must be sequentially extracted. These two restrictions make us to propose the following learning algorithm:

Constructive Oriented PCNN Learning Algorithm (*m, T', V', Y,* $\mathrm{L}_{\text{classifier}}(\Psi(\bullet))$*, l, b, A)*→ ***U***

m is the number of OPCs to extract

T'= $\{\mathbf{x}_i', \quad i = 0...\mathrm{N}-1\}$ is the training set

Val'= $\{\mathbf{x}_i^{\mathrm{V}}', \quad i = 0...\mathrm{N}_{\mathrm{V}}-1\}$ is the validation set

Ψ is the classifier

λ is the regularisation parameter

b is the number of epochs to iterate before we check the validation error

A is the initial value of matrix **U** for the adaptive algorithm

U is the linear neural network whose weights are the first m OPCs

Step 0. Whitening of T' and Val':

Step 0.1. Computation of the training sample mean $\langle \mathbf{x} \rangle = \frac{1}{\mathrm{N}} \sum_{i=0}^{N-1} \mathbf{x}_i'$

Step 0.2. Removal of mean in T' and Val':

T= $\{\mathbf{x}_i = \mathbf{x}_i' - \langle \mathbf{x} \rangle, \quad i = 0...\mathrm{N}-1\}$, Val= $\{\mathbf{x}_i^{\mathrm{V}} = \mathbf{x}_i^{\mathrm{V}}' - \langle \mathbf{x} \rangle, \quad i = 0...\mathrm{N}_{\mathrm{V}}-1\}$

Step 1. Initialisation of $\mathbf{U}[0] = \mathbf{A} = [\mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_m]$

Step 2. Estimation of m PCs in a sequential fashion.

Step 2.1. k=1.

Step 2.2. n=0.

Step 2.3. Initialisation of $\mathbf{V}^k[0] = [\mathbf{v}_1^* \quad \mathbf{v}_2^* \quad \cdots \quad \mathbf{v}_{k-1}^* \quad \mathbf{v}_k[0]]$ where $\mathbf{v}_k[0] = \mathbf{a}_k$

Step 2.4. Instantaneous estimation of k[th] PC

Step 2.4.1. Adaptation of $\mathbf{V}^k[n] = [\mathbf{v}_1^* \quad \mathbf{v}_2^* \quad \cdots \quad \mathbf{v}_k[n]]$

$$\mathbf{v}_k'[n+1] = \mathbf{v}_k[n] - h[n]\left(\frac{\partial \mathrm{L}_{\mathrm{PCA}}[n]}{\partial \mathbf{v}_k[n]} + l\,\frac{\partial \mathrm{L}_{\text{classifier}}[n]}{\partial \mathbf{v}_k[n]}\right) = \tag{17a}$$

$$= \mathbf{v}_k[n] + h[n]\left(\mathbf{e}[n]^{\mathrm{T}}\mathbf{v}_k[n]\mathbf{x}[n] + \mathbf{v}_k[n]^{\mathrm{T}}\mathbf{x}[n]\mathbf{e}[n] + l\,\frac{\partial \mathrm{L}_{\text{classifier}}[n]}{\partial \mathbf{v}_k[n]}\right)$$

$$\vec{\mathrm{e}}[n] = \vec{\mathrm{x}}[n] - \mathbf{V}^k[n]\mathbf{V}^k[n]^{\mathrm{T}}\mathbf{x}[n] \tag{17b}$$

$$\mathbf{V}^k[n] = [\mathbf{v}_1^* \quad \mathbf{v}_2^* \quad \cdots \quad \mathbf{v}_k[n]] \tag{17c}$$

$$\mathbf{v}_k[n+1] = \frac{\mathbf{v}_k'[n+1]}{\left\|\mathbf{v}_k'[n+1]\right\|} \tag{17d}$$

Where $\mathbf{x}[n] = \mathbf{x}_{n \bmod N}$, $\mathrm{L}_{\mathrm{PCA}}[n]$ is the instantaneous reconstruction error given by

$$L_{PCA}[n] = \frac{1}{2}\left\|\mathbf{x}[n] - \mathbf{V}^k[n]\mathbf{V}^k[n]^T\mathbf{x}[n]\right\|^2 \tag{18}$$

And $L_{classifier}[n]$ is the instantaneous penalty term,

$$L_{classifier}[n] = L_{classifier}\left(\Psi\left(\mathbf{U}[n]^T\mathbf{x}[n]\right)\right) \tag{19a}$$

$$\mathbf{U}[n] = \left[\mathbf{v}_1^* \quad \cdots \quad \mathbf{v}_{k-1}^* \quad \mathbf{v}_k[n] \quad \mathbf{a}_{k+1} \quad \cdots \quad \mathbf{a}_m\right] \tag{19b}$$

Step 2.5. **If** (n mod b)=0 **then**

Step 2.5.1. Evaluate the average validation error,

$$\langle E_V \rangle = \frac{1}{N_v}\sum_{i=0}^{N_V-1}\left(\left\|\mathbf{x}_i^V - \mathbf{V}^k[n]\mathbf{V}^k[n]^T\mathbf{x}_i^V\right\|^2 + l\ L_{classifier}\left(\Psi\left(\mathbf{U}[n]^T\mathbf{x}_i^V\right)\right)\right) \tag{20}$$

Where $\mathbf{V}^k[n]$ and $\mathbf{U}[n]$ are given by equations (17c) and (19b) respectively .

Step 2.5.2. **If** $\langle E_V \rangle$ increases **then** $\mathbf{v}_k^* = \mathbf{v}_k[n]$ and **go to** Step 2.8.

Step 2.6. n=n+1

Step 2.7. **Go to** Step 2.4.

Step 2.8. k=k+1

Step 2.9. **If** k>m **then go to** Step 3.

Step 2.10. **Go to** Step 2.2.

Step 3. $\mathbf{U} = \mathbf{V}^m = \left[\mathbf{v}_1^* \quad \mathbf{v}_2^* \quad \cdots \quad \mathbf{v}_m^*\right]$

Step 4. End

Next we will prove that this algorithm effectively estimates the first m PCs in the high-variance sense when λ=0. The algorithm starts computing the first component $\mathbf{v}_1$. Since we perform an on-line gradient descent over the average reconstruction error subject to the constraint $\|\mathbf{v}_1[n]\| = 1$ (equation 17c), the sequence $\{\mathbf{v}_1[n], \quad n \geq 0\}$ tends with probability one when $N \rightarrow \infty$ to the PCs with the highest variance. Another way of seeing this, it is that the learning equation (17) is equivalent to the simplified version of Oja's Rule which asymptotically converges to the first PC (p.370-374, Haykin, 1994).

Once we estimate the first PC, we start to compute the second component, freezing the achieved value of the first component and minimising the average reconstruction error when the dimension of the feature space is two. Again, the solution of this problem is an estimation of the two first PCs or a linear combination of them. Nevertheless, since the first component is fixed to

the first estimated PC, the only possible solution to the optimisation problem is that the computed second component is an estimation of the second PC. Then by induction we can prove that the proposed constructive learning algorithm for λ=0 estimates the first m PCs.

4. The Global Learning Algorithm

According to the last section, we can use the constructive OPCNN with any maximum classifier that admits the computation of $\partial L_{\text{classifier}}(\Psi, \mathbf{U})/\partial \mathbf{U}$. However, the function $L_{\text{classifier}}(\Psi, \mathbf{U})$ must be chosen in order to be a good candidate for a gradient search. Since the average number of misclassification (equation 14) is obviously a poor candidate, we need to use better choices like the cross-entropy loss function (equation 11). In fact, the most reasonable election might be closely related with the loss function that uses the classifier in its learning.

We next review the basic co-operative Learning Algorithm for OPCA + classification presented in section 2 introducing a criterion to finish the learning process based on the early stopping technique. Suppose we can call a learning procedure for the classification system given by

Classifier's Learning Algorithm ***(****T, V,* ***U****,* $\mathbf{Y}_{ini}$***)*** $\rightarrow$ ***Y***

where T is the training set, V is the validation set, **U** is the linear transformation that uses the classifier for pre-processing, $\mathbf{Y}_{ini}$ is a initial configuration of the classifier and, Ψ is the resulting classifier that minimises e.g. the empirical misclassification error computed using the validation set. Then the global learning algorithm can be written as follows:

Global Learning Algorithm for OPCA+Classification (*m, T, V,* ***Y****,* $L_{\text{classifier}}(\Psi(\bullet)), l, b$***)***

m is the number of OPCs to extract

T= $\{(\mathbf{x}_i, \mathbf{y}_i), \quad i = 0...N-1\}$ is the training set

V= $\{(\mathbf{x}_i^V, \mathbf{y}_i^V), \quad i = 0...N_V - 1\}$ is the validation set

Ψ is the classifier

λ is the regularisation parameter

b is the number of epochs to iterate before we check the validation error in the OPCNN learning algorithm

Step 1. $\mathbf{U^0=0}$

Step 2. k=1

Step 3. **If** k=1 **then** $\lambda[k]=0$ **else** $\lambda[k]=\lambda$

Step 4. Compute the OPCs:

Constructive Oriented PCNN Learning Algorithm ***(****T, V,* $\mathbf{Y}^{k-1}$, $L_{classifier}(\Psi(\bullet))$, *l[k], b,* $\mathbf{U}^{k-1}$***)*** $\rightarrow$ $\mathbf{U}^{k}$

Step 5. Compute the classifier $\mathbf{Y}^{k}$ that uses the linear transformation $\mathbf{U}^{k}$

Classifier's Learning Algorithm ***(****T, V,* $\mathbf{U}^{k}$, $\mathbf{Y}^{k-1}$***)*** $\rightarrow$ $\mathbf{Y}^{k}$

Step 6. Evaluate the average misclassification error of the overall system using the validation set,

$$L_{\text{pattern recogniser}} = \frac{1}{N_V}\sum_{i=0}^{N_V-1} 1\left(\Psi^k\left(\mathbf{U}^{k\,T}\mathbf{x}_i^V\right) \neq \text{class label}\left(\mathbf{x}_i^V\right)\right) \tag{21}$$

Step 7. **If** $L_{\text{pattern recogniser}}$ increases **then go to** Step 10.

Step 8. k=k+1

Step 9. **Go to** Step 3

Step 10. End

5. The adaptive Soft k-Nearest-Neighbour Classifier

As we have pointed out in several places of this paper, we need a classifier in order to apply the global training displayed in the above section. The only restriction is that it must allow the computation of $\partial L_{classifier}(\Psi,\mathbf{U})/\partial\mathbf{U}$. In this section, we introduce the adaptive soft K-nearest-neighbour (K-NN) classifiers (Bermejo & Cabestany, 1999) that fulfil this condition and besides they have interesting properties like:

- Better approximation capabilities than crisp K-NN classifiers
- The use of fuzzy LVQ-like learning procedures with faster convergence speed than Kohonen's LVQ algorithms in some particular cases.

The so-called adaptive soft K-NN classifiers are local learning algorithms (Bottou & Vapnik, 1992) based on the idea of enhancing the approximation qualities of K-NN estimation by means of the use of a local Parzen window. This local window is defined in a hypersphere centred at the pattern to classify **x** that contains precisely the k nearest neighbours prototypes. Their discriminant functions are given by

$$d_l^*(\mathbf{x};g) = \frac{\sum_{i=1}^{K_l} G\left(\mathbf{x}_i^{\,l} - \mathbf{x};g\right)}{\sum_{j=1}^{C}\sum_{u=1}^{K_j} G\left(\mathbf{x}_u^{\,j} - \mathbf{x};g\right)}, \quad l = 1,\ldots,C \tag{22}$$

where kernel G is a bounded and even function on X that is peaked about **0,** γ denotes the locality control parameters associated to G, $\{\mathbf{x}_u^j, u = 1,...,K_j\}$ are those prototypes of class j among the K nearest prototypes to **x** and $K = \sum_{j=1}^{C} K_j$. As any maximum classifier, they assign a new input to that class whose discriminant function has the highest value.

If the set of prototypes are constrained to coincide with the training points, these discriminant functions are posterior class estimates and also the solution of a local constant weighted least squares regression problem that involves the training data. Otherwise, the local kernel estimators turn into a local mixture model in which the prototypes are the centres of a parameterised density function G. (Of course, the kernel G must satisfy the conditions of a probability density function.) In this latter case, a learning algorithm must estimate the parameters of the mixture model from data.

Suppose we restrict the mixture model parameters to centres and a global defined parameter σ that controls locality. Then the goal of the learning phase can be reduced to compute a series of set of prototypes (or codebooks) $\mathbf{C_j}=\{\mathbf{w}_i^j, i = 1,...,M_j\}$ j=1,...,C, one for each class, that minimises some loss function related to the misclassification rate. We propose to minimise the following modified cross-entropy error function:

$$L = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{l=1}^{C} y_{li}\, d_l(\mathbf{z}_i) \tag{23}$$

where $\mathbf{z_i}$ is, in this case, the linearly transformed input point $\mathbf{U^T x_i}$. This function has the same minimum point as the usual cross-entropy function (equation 11) but besides it has two remarkable properties:

- If class overlapping is moderate and the number of training samples tends to infinite, the absolute value of the minimum of L tends to the probability of correct classification.
- Outliers' impact is reduced in a gradient descent approach to learning

Instead of using this modified cross-entropy loss function (equation 23) as the penalty term in the constructive oriented PCNN learning algorithm, we will make use of an a positive-defined error function that has the same minimum points and gradient information:

$$
\begin{aligned}
L_{\text{classifier}}(\Psi, \mathbf{U}) &= \frac{1}{N}\sum_{i=0}^{N-1} L^{i}_{\text{classifier}}(\Psi, \mathbf{U}) \\
L^{i}_{\text{classifier}}(\Psi, \mathbf{U}) &= 1 - d_j\left(\mathbf{U}^T \mathbf{x}_i\right); \quad j = \text{class label}(\mathbf{x}_i)
\end{aligned}
\tag{24}
$$

For analytical convenience, G is a gaussian kernel of the form $1/(2p\sqrt{s})^m \exp(\bullet/s)$. We also multiply equation (24) by the factor $s/2$ to eliminate this constant in the learning equations. Then the partial derivative of $L^{i}_{\text{classifier}}(\Psi, \mathbf{U})$ respect to the k^{th} column vector of **U** yields

$$
\begin{aligned}
\frac{\partial L^{i}_{\text{classifier}}}{\partial \mathbf{u}_k} = & \frac{\sum_{l=1}^{K_j} \left(\vec{u}_k{}^T \mathbf{x}_i - w_{ik}{}^{j}\right) \exp\left(-\left\|\mathbf{x}_i - \mathbf{w}_l{}^{j}\right\|^2 / s\right)}{\sum_{m'=1}^{C}\sum_{u=1}^{K_{m'}} \exp\left(-\left\|\mathbf{x}_i - \mathbf{w}_u{}^{m'}\right\|^2 / s\right)} \mathbf{x}_i - \\
& - d_j(\mathbf{x}_i) \frac{\sum_{m'=1}^{C}\sum_{u=1}^{K_{m'}} \left(\vec{u}_k{}^T \mathbf{x}_i - w_{uk}{}^{m'}\right) \exp\left(-\left\|\mathbf{x}_i - \mathbf{w}_u{}^{m'}\right\|^2 / s\right)}{\sum_{m'=1}^{C}\sum_{u=1}^{K_m} \exp\left(-\left\|\mathbf{x}_i - \mathbf{w}_u{}^{m'}\right\|^2 / s\right)} \mathbf{x}_i
\end{aligned}
\tag{25a}
$$

$$
d_j(\mathbf{x}) = \frac{\sum_{l=1}^{K_j} \exp\left(-\left\|\mathbf{x} - \mathbf{w}_l{}^{j}\right\|^2 / s\right)}{\sum_{m'=1}^{C}\sum_{u=1}^{K_m} \exp\left(-\left\|\mathbf{x} - \mathbf{w}_u{}^{m'}\right\|^2 / s\right)}
\tag{25b}
$$

6. Experimental Results

One must expect better performance of our procedure compared to PCA in classification problems where the separation between classes is not in the direction of high-variance PCs. To get some insight into how OPCA is performed, we present experiments with two artificial problems where low-variance PCs are the right discriminate directions.

In real-life data, underlying class densities are unknown. In this way, comparative studies on this kind of data will tend to be less informative since we do not know in advance if PCA will be enough to discriminate between classes. However, we also include simulations with real-data to see how the learning system behaves in a real-world problem.

6.1. The 2-D Problem

This artificial example was constructed to be completely unfavourable to PCA based on extracting the principal component of highest variance. There are two 2-D normal classes with

the same priors. The data for the first class was generated from a normal distribution $\mathbf{x_1} \sim \mathbf{N}(\mathbf{m_1},\mathbf{K})$ with the class mean and covariance matrix given by

$$\mathbf{m}_1 = \begin{bmatrix} 0 \\ 1.5 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \tag{26}$$

For the other class, data was sampled from a normal distribution $\mathbf{x_2} \sim \mathbf{N}(\mathbf{m_2},\mathbf{K})$with the same covariance matrix than before and the class mean given by

$$\mathbf{m}_2 = \begin{bmatrix} 0 \\ -1.5 \end{bmatrix} \tag{27}$$

The goal here is to build a pattern recogniser using a feature extractor that projects the original space into a line. As one can infer from figure 1, the component of lowest variance is the optimal linear transformation for this classification problem. So one can expect that the optimal oriented PC will be for a value of $\lambda>0$ and will coincide with the lowest variance PC.

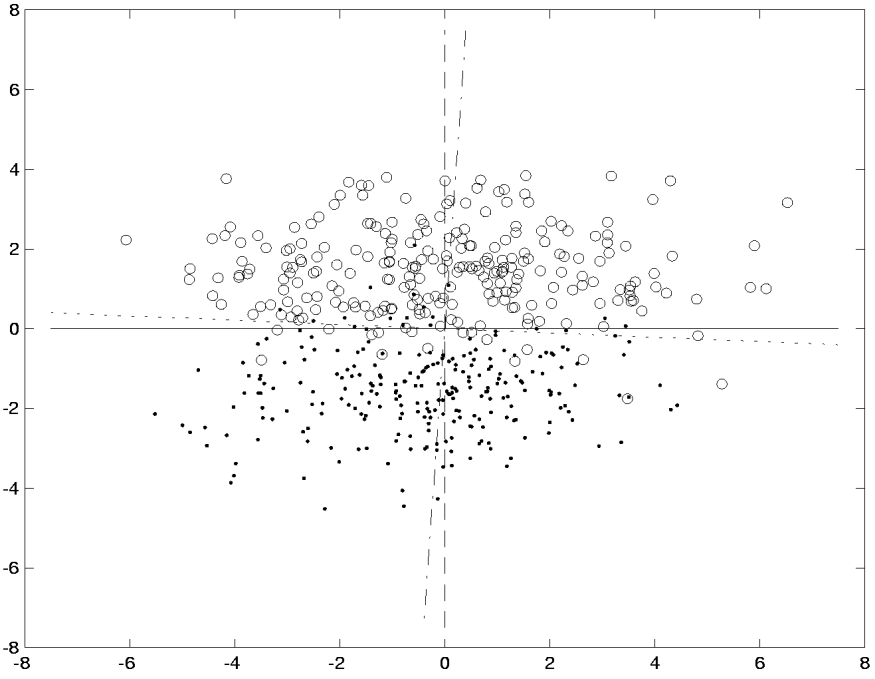


Fig. 1. The 2-D problem and its principal components. We show data samples from class 1 (labelled 'O') and class 2 ('.'), the optimal PCs- the dashed (--) and solid (-) lines- and the sample PCs- the dotted (.) and dashdot (-.) lines- that are estimated from training data. Observe that due to finite sample size the estimated PCs are slightly rotated versions of PCs.

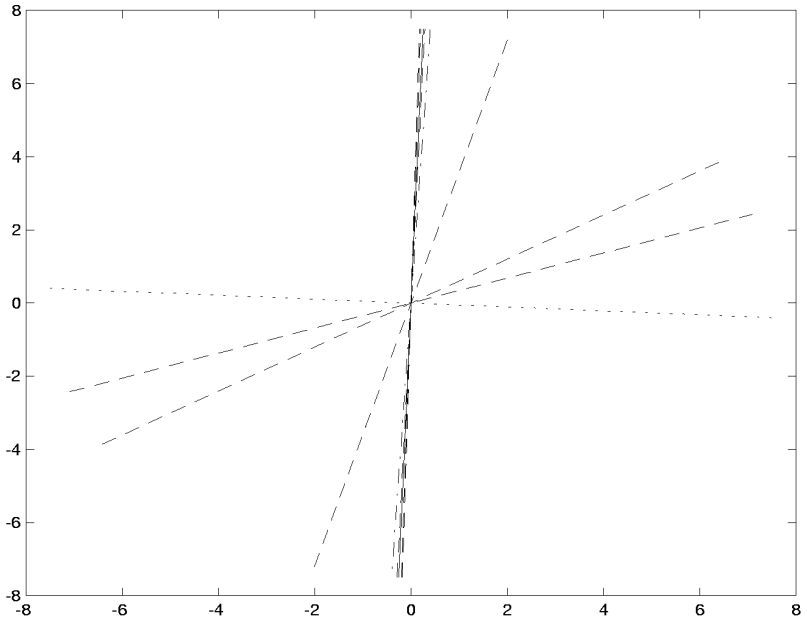


Fig. 2. The Oriented PCs and sample PCs in the ninth set of the 2-D problem. We show the sample PCs, the dotted (.) and dashdot (-.) lines. These have been estimated from the sample covariance matrix formed with the 9^{th} training data. We also display the oriented PCs, the dashed (--) lines, for values of λ=1, ..., 90 (the estimated optimal value). As λ increases, the Oriented PC goes from the estimated PC of highest variance to the estimated PC of lowest variance.

Ten independent training, test and validation sets (of the same size, $N_{2\text{-}D}$=1000) are produced from the 2-D problem. Then one hundred runs for each training set using several values of λ (1, 2.5, 5, 7.5, 10, 15, 20, 30, 50, 70, 90, 100, 200, 250, 300, 400, 500) have been made. The average misclassification rate for each test set is computed over $1000N_{2\text{-}D}$ classifications. We have initialised the set of prototypes of the soft 2-NN classifier using LVQ_PAK's eveninit program (Kohonen, 1995) which selects randomly from training data an even number of prototypes that fall inside class borders. The optimal value of the parameters has been estimated using the validation set.

Figure 3 shows the results for the ten test sets. The overall average misclassification rate (averaged over the patterns of the ten test sets) decreases from 46.99 (λ=1) to 9.78 (λ=90). As we display on figure 2 as λ increases, the oriented PC goes from the sample PC of highest variance to the sample PC of lowest variance.

The Bayes decision regions are separated by the line y=0 so the original problem can be solved in a one dimensional space. Using the Bayes classifier, the probability of

misclassification $P(E_B)$ is $1/2\,\mathrm{erfc}\left(1.5/\sqrt{2}\right)=0.0668$ where erfc is the complementary error function. For this problem we have shown that the regularisation process moves OPC from the high-variance PC to the low-variance PC. Therefore, the average misclassification error might goes from $1/2\,\mathrm{erfc}(0)=0.5$ (for λ=0) to $1/2\,\mathrm{erfc}\left(1.5/\sqrt{2}\right)=0.0668$ (for the optimal λ). As we have shown the best classification result was 9.78%, more than three percent units greater than $P(E_B)$. This behaviour of the overall recognition system has an easy explanation. To optimally resolve the 2-D classification problem in a linearly transformed feature space, the feature extractor u must coincide with the y-axis. This does not happen due to the sample size effect since the sample PC of smallest variance does not coincide with the y-axis (see figures 2 and 3). Hence the resulting 1-D classification problem has a higher $P(E_B)$ than 6.68%.

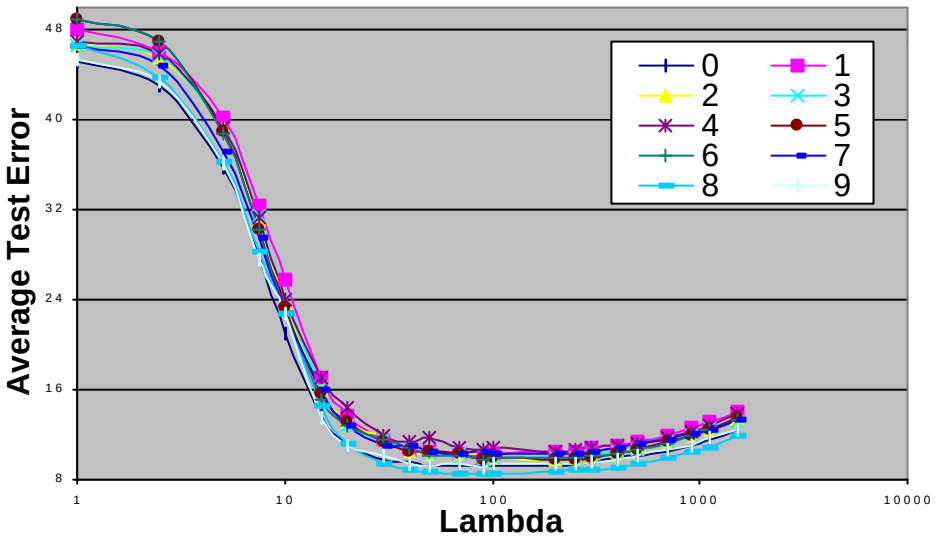


Fig. 3. The average misclassification rate of each test set of the 2-D Problem versus lambda. Note that as lambda increases the test error decreases until reaching a minimum. Then the test error starts to moderately increase.

6.2. The 3-D Problem

This second artificial example is again a problem that can be solved using only the first PC of smallest variance. There are two 3-D normal classes with the same prior class probabilities. The data for the first class was extracted from a normal distribution $\mathbf{x_1} \sim N(\mathbf{m_1},\mathbf{K})$ with the class mean and covariance matrix given by

$$\mathbf{m}_1 = \begin{bmatrix} 0 \\ 1.5 \\ -1.5 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 7 \end{bmatrix} \tag{28}$$

For the second class, data was generated from a normal distribution $\mathbf{x_2} \sim N(\mathbf{m_2}, \mathbf{K})$ with the same covariance matrix **K** and the class mean given by

$$\mathbf{m}_2 = \begin{bmatrix} 0 \\ -1.5 \\ 1.5 \end{bmatrix} \tag{29}$$

Instead of building a 1-OPC Analyser plus a 1-D classifier, we force to resolve the 3-D problem in a 2-D feature space in order to test the extraction process of multiple PCs in a redundant feature space.

Again, ten independent training, test and validation sets (of the same size, $N_{3\text{-}D}$=1000) are produced from the **3-D** problem. Then one hundred runs for each training and several values of λ (1, 2.5, 5, 7.5, 10, 15, 20, 30, 50, 70, 90, 100, 200, 250, 300, 400, 500, 700, 900, 1100, 1500) have been made. The average misclassification rate for each test set is computed over $1000N_{3\text{-}D}$ classifications. We have initialised the set of prototypes of the classifier and estimate the optimal value of the parameters as before using the soft 2-NN classifier.

Figure 4 shows the results for the ten test sets. The overall average misclassification rate decreases from 26.58% (λ=1) to 16.55% (λ=500). For this problem, the Bayes decision regions are separated by the plane y=0. Then the minimum probability of misclassification is $1/2\,\mathrm{erfc}(1.5/2) = 0.1444$. If the regularisation process moved the OPCs from high-variance PCs to low-variance PCs, we could observe that the average misclassification error would go from approximately $1/2\,\mathrm{erfc}(1.5/\sqrt{14}) = 0.2854$ (for λ=0) to $1/2\,\mathrm{erfc}(1.5/2) = 0.1444$ (for the optimal λ). In fact, this is the empirically observed fluctuation. However, which is the real solution achieved for the optimal lambda? Does the two OPCs coincide with the sample PCs of lowest variance?

In figures 5 and 6, we can observe the computed OPCs in 1000 different runs of the learning algorithm using the ninth training set. Initially, for low values of λ, the OPCs departure from high-variance PCs (the z-axis and the x-axis). As λ increases, the first OPCs tend in average to the low-variance PC (the y-axis). However, the second OPCs do not converge to the second

low-variance PC but their location tend to be uniformly distributed over the surface of the sphere of radius 1. Although there is a 2-D feature space, the classifier can solve the classification problem in one dimension. Hence, the second OPC is redundant so the learning algorithm randomly chooses its value. In consequence, the overall system computes those optimal directions to project data to classify but using only the minimum number of projections to effectively solve the problem.

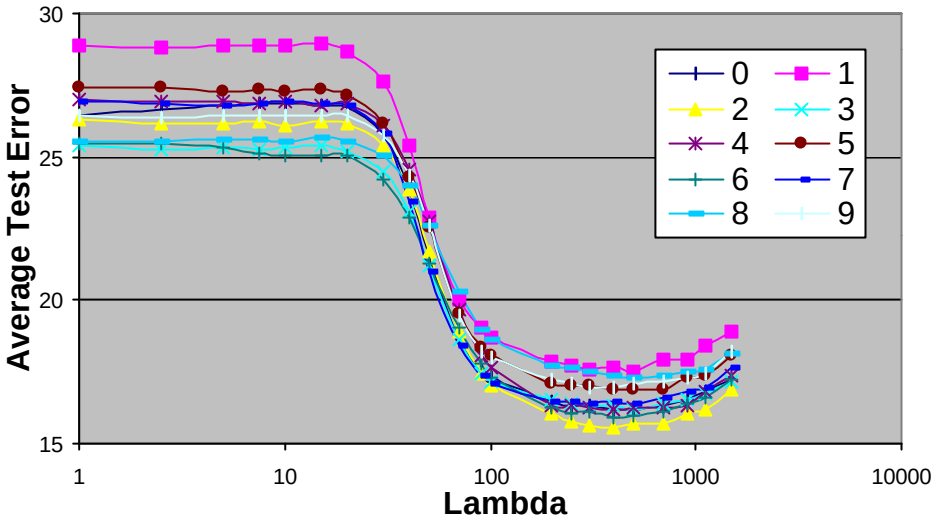


Fig 4. The average misclassification rate of each test set versus lambda in the 3-D Problem. Notice that, like in the 2D-problem, as lambda increases the test error decreases until reaching a minimum. Then the test error starts to moderately increase.

6.3. The Satimage Problem

The Satimage database was taken originally from the UCI Repository of machine learning databases (Murphy & Aha, 1994). The Australian Centre for Remote Sensing generated it from Landsat Multi-Spectral Scanner image data from NASA. The database contains 6435 patterns with 36 components belonging to five classes.

We have divided the original data in training, test, and validation sets that have the following sizes: 3217, 2146 and 1072 respectively. Then ten runs for different sizes of the set of prototypes (6, 12, 24, 48, 96, and 192), several values of lambda (0,5,10,15,30,50,70,100,200), and m, the number of extracted OPCs , (1, 2, 4, 6, 8, 10, 12, 14, 16, and 18) have been made. We have initialised the set of prototypes and have estimated the optimal value of the parameters as before with the exception of the adaptive soft 2-NN classifier's parameter sigma. We estimated with the

validation set an optimal value when m=1. We then apply this value for all other cases. This is done to see how the performance is degraded as m increases since a wrong value of sigma can considerably deteriorate the gradient information that the classifier gives to the OPCNN. The result of this loss of information can notably affect the performance of the overall system that could fail to solve the problem.

In figure 7, we display the average misclassification rate of the test set as function of m, for each size of the set of prototypes. We can observe that given a fixed number of the set of prototypes, OPCA is always useful. There is always a certain range of λ in which OPCs achieves better separation between classes than PCs. Besides, the optimal number than OPCS are smaller than the optimal number of PC. Hence, the classifier's complexity is reduced using OPCA in front of PCA and the resulting pattern recogniser can achieve a better generalisation performance.

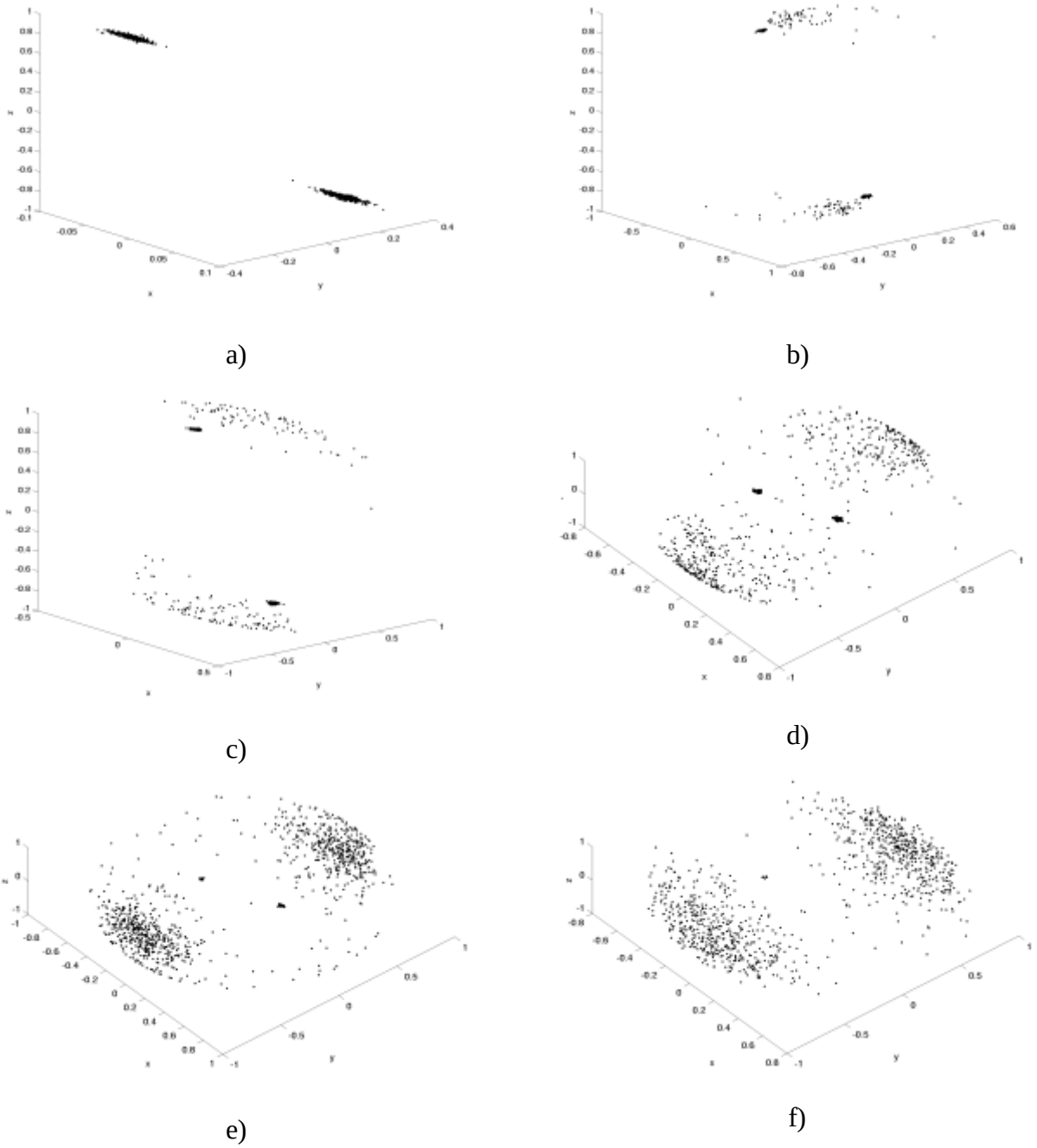
Finally, another interesting result is that the degradation, for using a sub-optimal value of σ, in the classification accuracy of the overall system as m and the number of prototypes increase is moderate. In table 1, we show the classification error of 1-nearest-neighbourg classifier as a function of extracted principal components. One can see that our combined OPCA+classifier system outperforms the PCA+1-NN classifier for low-dimensional feature spaces (m=1 and m=2) as expected (table 2). When the dimension of the feature space increases, the most remarkable effect is that the use of more OPCs does not provoke a great benefit in the classification accuracy and neither a great catastrophe in the classification accuracy.

Dim	1	2	4	6	8	10	12	14	16	18
Error %	54.94	22.11	14.25	11.17	11.10	11.17	11.69	11.81	12.39	13.92

Table 1. Misclassification rate of a 1-nearest neighbour classifier as a function of the principal components extracted for the Satimage databases. (These results are taken from ELENA project (p.28; Jutten et al.,1995)).

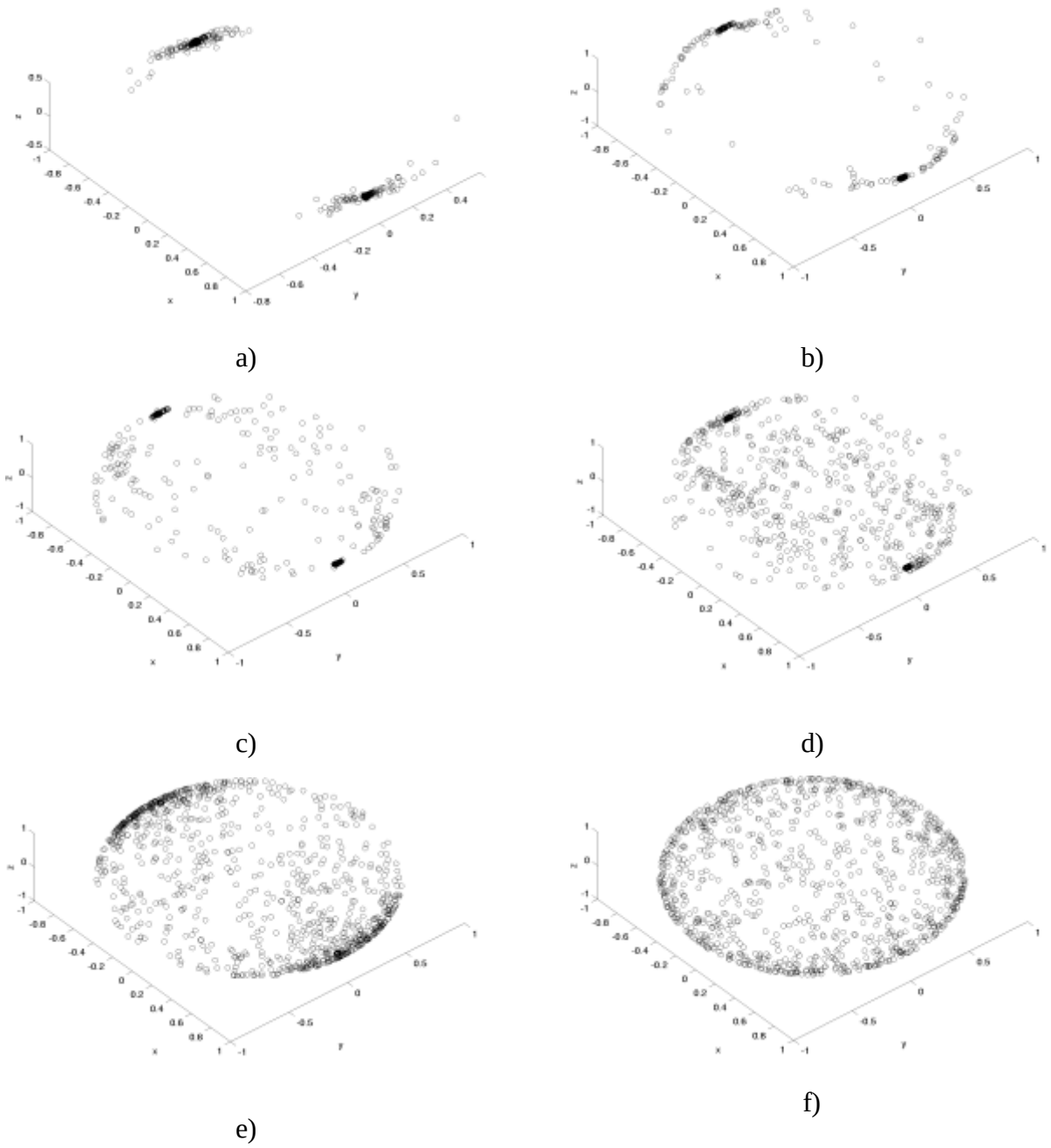
Dim	1	2	4	6	8	10	12	14	16	18
Err%	47.91	17.09	15.06	14.26	14.29	14.24	14.67	14.49	14,83	14,60

Table 2. Lower Average Misclassification rate classifier as a function of the principal components extracted for OPCA+ soft k-NN classifier with 192 prototypes.



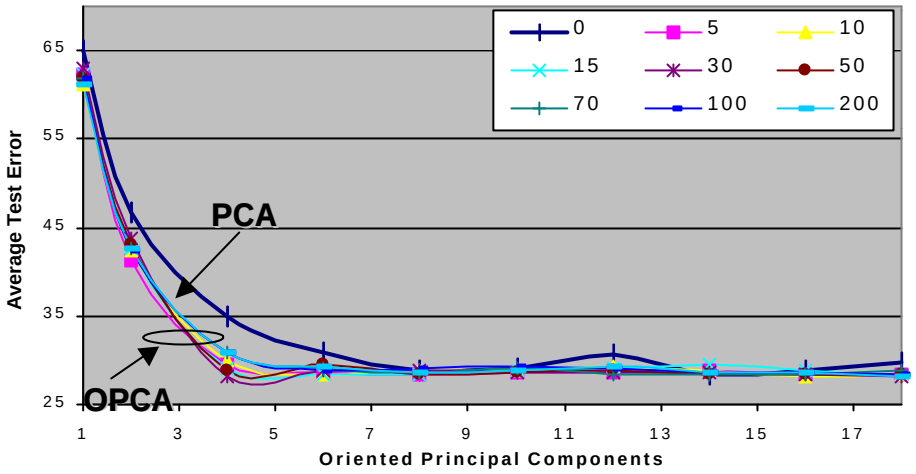
a) b) c) d) e) f)

Fig. 5. Evolution of the computed first oriented principal components in 1000 runs as lambda increases using the ninth training set of the 3-D problem. a) λ=1, b) λ=20, c) λ=30, d) λ=50, e) λ=100, f) λ=300. For λ=1, the first OPCs converge to the highest-variance PC (the z-axis). As lambda increases, the original solution softly vanishes and the final solution starts to appear. (For instance this is notoriously evident for λ=50 where the two solutions coexist.) For λ=300, the first OPCs are located in the region of the lowest-variance PC (y-axis).

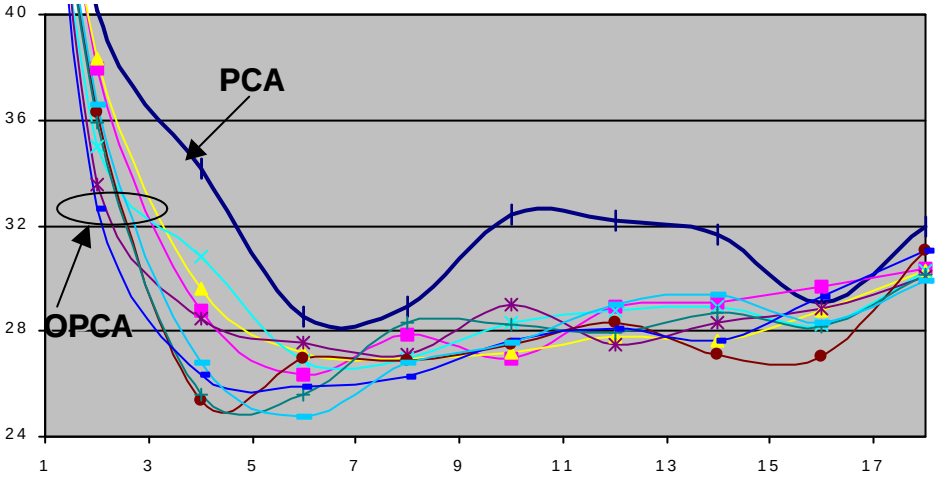


a) b) c) d) e) f)

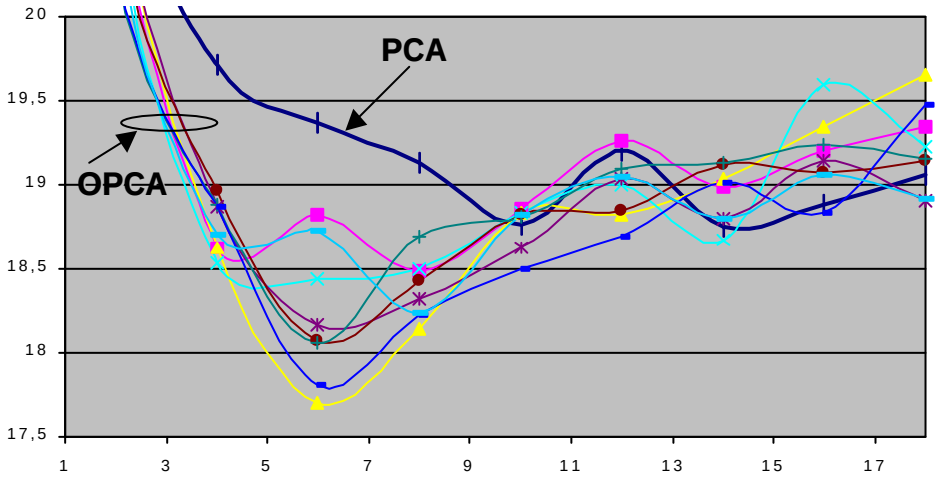
Fig. 6. Evolution of the computed second oriented principal components in 1000 runs as lambda increases using the ninth training set of the 3-D problem. a) λ=10, b) λ=20, c) λ=30, λ=50, d) λ=100, e) λ=300. For λ=1, the second OPC is located in the region of the second highest-variance PC (the x-axis). As lambda increases, the original solution is softly disintegrated. Since the pattern classifier only needs 1-OPC to solve the problem, as the first OPC converges to the optimal solution the second OPC is not longer constrained to any region of the surface of the sphere. Hence, for λ=300 the distribution of second OPCs is uniformly located over the surface.


0
5
10
15
30
50
70
100
200
Average Test Error
65
55
45
35
25
PCA
OPCA
1
3
5
7
9
11
13
15
17
Oriented Principal Components

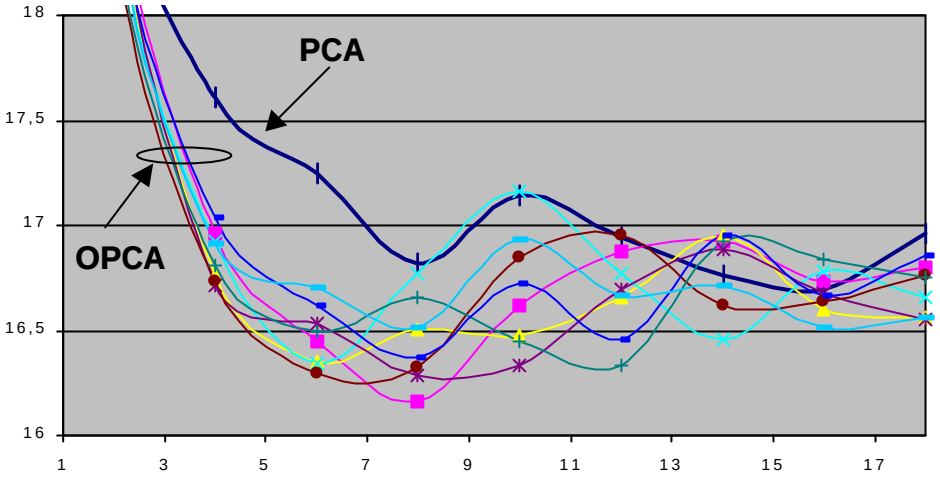
a)


40
36
32
28
24
PCA
OPCA
1
3
5
7
9
11
13
15
17

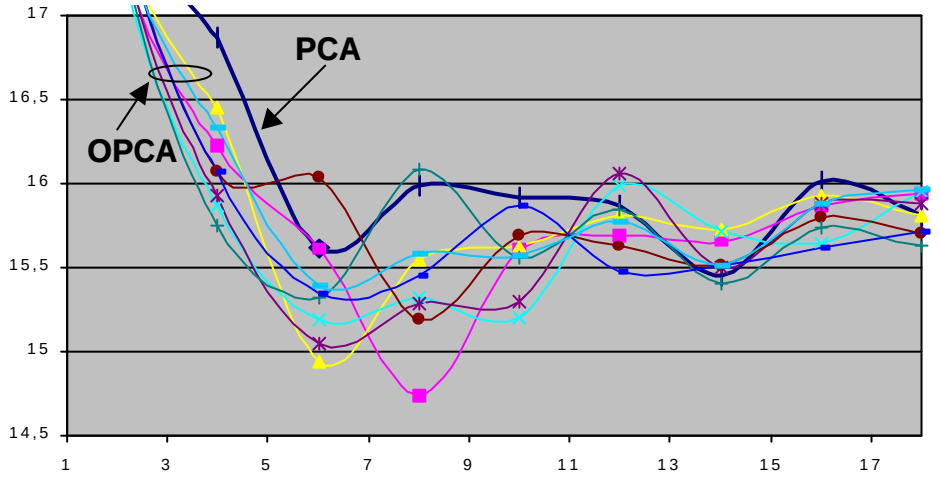
b)


20
19,5
19
18,5
18
17,5
PCA
OPCA
1
3
5
7
9
11
13
15
17

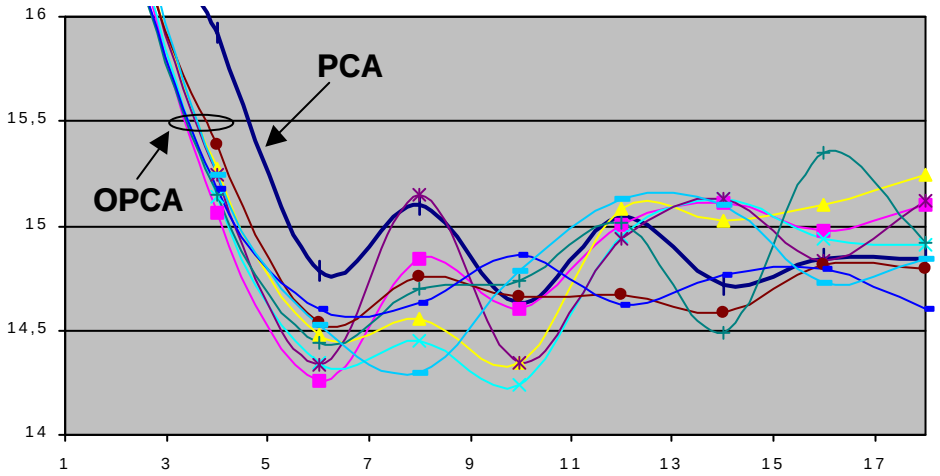
c)



d)



e)



f)

Fig 7 . The average test misclassification rate versus λ in the Satimage Problem for different sizes of the set of prototypes: a) 6, b) 12, c) 24, d) 48, e) 96, f) 192. We show in a thick line the

solution achieved by the pattern recogniser based on PCA and in thin lines the solutions based on OPCA. Note that given a fixed number of classifier's prototypes, OPCA is always useful since the pattern recognisers that use OPCs achieve better recognition accuracy than those based on PCA for a certain range of the regularisation parameter λ. Another important result is that the number of OPCs for the best λ is always smaller than the number of PCs needed to achieve a minimum of the test error.

6. Discussion

6.1. Co-operative feature extraction and classification.

Feature extraction/selection techniques form linear (or non-linear) combinations of the original variables of the same size and then select the most relevant combination. Clearly, the optimal subset of features selected and the transformation to compute these features depend on the classification problem and the particular classifier with which they are used (p.304, Bishop, 1995). However, the traditional design process of a pattern recogniser is based on training each module individually. Hence, the use of unsupervised techniques in feature extraction is common but since they take no account of class information it is difficult to predict how well they will work as pre-processing for a given classification problem. In some cases, the representation of the input space through an economic description (compression) can involve a vital loss of information to classify. (For instance, PCA is based on projecting the input space using the directions in which data varies most in order to reconstruct the original point from the transformed space with the minimum error. Although, as we have just seen in the last section, the directions of lowest-variance are vital in some classification problems.) Therefore, it is necessary to integrate feature extraction in the classification process. This implies to design the pattern recogniser as a whole as other works have been already proposed (e.g. (LeCun et al., 1999) (Bottou&Gallinari,1991)).

We have introduced in this paper a well-principled design paradigm based on coupling the learning algorithms of a PCNN and a classifier in a global process. The resulting global learning algorithm searches a feature space based on a linear transformation useful for class discrimination according to the influence of the feedback signals of the classifier in the linear neural network. This iterative regularisation process produces a rotation of the original high-variance PCs controlled by the parameter λ. As we have just seen in the simulations when λ increases, the OPCs converge to those projections that achieve a better separation of points in the feature space.

6.2. Learning to learn the feature space.

In our approach the initial solution are reached through separate feature extraction and classification. Then we compute a new feature extractor based on the misclassification error function of this first solution and re-train the classifier again using the new feature extractor. We restart the design process using now the last solution as the initial condition for new pattern recogniser. We end this iterative process when the misclassification error on the validation set stops decreasing. These iterative and coupled learning procedures converge to a stable solution that reflect the mutual constraints between the feature extractor and the classifier in order to reach optimal classification accuracy.

This global learning algorithm belong to the group of the 'learning to learn' algorithms (Thrun and Pratt, 1998) that learn constraints which are superposed when learning a new pattern recogniser. Our system learns to constrain the linear transformation to those directions where a cost function, composed of a fixed PCA-based error plus a variable term that is a function of the last learned recogniser, is minimised. So we use a previously learned pattern recogniser to train the new system and to constrain the feature space to those subspaces in which classification and compression and properly balanced according to the value of λ.

6.3. OPCA and Bayes VQ.

We have emphasised the use of OPCA as a useful technique for classification purposes. However, it can be employed for mixed compression/classification purposes. See for instance work on Bayes Vector Quantization (Gray & Olsen, 1997) (Perlmutter et al., 1996) (Oehler & Gray, 1995). In Bayes VQ the emphasis is to explore the combined data compression/classification goal. The synergistic Bayes VQ design also incorporates a classification term into the distortion measure by adding a penalising weighted term to the usual cost function for compression purposes. Depending on the value of the weight λ, the emphasis will be more on compression or on classification.

7. Conclusions

OPCA is presented as a well-principled technique for feature extraction in classification problems involving data that belong to several classes. It incorporates class information by adding a regularising term in the usual PCA average reconstruction error. Depending on the value of the regularisation parameter λ, the oriented PCs are somewhere in between high-variance PC and those directions that maximise class separation in the feature space. A

constructive oriented principal component neural network is introduced to estimate in an adaptive fashion the OPCs from data.

In order to compute the OPCs in an efficient way, a global learning algorithm for a pattern recogniser based on OPCA is presented. This global algorithm integrates the process of design of the feature extractor and the classifier into a single step. It builds the final system using a cascade of trained recognisers. The previously learned pattern recogniser is employed as the initial condition to train the new system and to constrain the feature space to those subspaces in which classification and compression and properly balanced according to the regularisation parameter λ.

Experimental results using artificial and real data show the potential of OPCA. Automatic detection of useful projections for classification is achieved. This means in practice to use less OPCs than PCs so the classifier works in a lower dimensional space and then generalisation performance is improved.

References

Bermejo, S., & Cabestany, J. (2000). Adaptive soft k-nearest neighbour classifiers. To be published *in Pattern Recognition, 33*.

Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford: Oxford University Press.

Bottou, L., & Vapnik, V. (1992). Local learning algorithms. Neural Computation, 4, 888-890.

Bottou, L. & Gallinari, P. (1991). A framework for the cooperation of learning algorithms. Lippmann, R. P., Moody, J. E. & Touretzky, D.S. (Eds.) Advances in Neural Information Processing Systems 3, Cambridge, MA: MIT Press.

Choe, Y., Sirosh, J., & Miikkulainen, R. (1996). Laterally interconnected self-organizing maps in hand-written digit recognition. Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (Eds.) Advances in Neural Information Processing Systems 8, Cambridge, MA: MIT Press.

Diamantaras, K.I., & Kung, S.Y. (1996). Principal component neural networks. Theory and applications. New York: John Wiley & Sons.

Fukunaga, K., & Koontz, W. C. G. (1970). Applications of the karhunen loeve expansion to feature extraction and ordering. IEEE Transactions on Computers, 19, 311-318.

Gray, R.M., & Olshen, R.A. (1997). Vector quantization and density estimation. Stanford, CA: Stanford University, Department of Electrical Engineering.

Haykin, S. (1994). Neural networks: a comprehensive foundation. New York: Macmillan College Publishing Company.

Jolliffe, I.T. (1986). Principal component analysis. New York: Springer-Verlag.

Jutten, C., Blayo, F. Cabestany, J., Cheneval, Y., Comon, P. & Verleysen, M. (Eds.) (1995). Elena Project. Esprit III Basic Research Action (No. 6891). Brussels: D Facto.

Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J., & Torkkola, K. (1995) LVQ_PAK. The Learning Vector Quantization Program Package (Version 3.1). Helsinki: Helsinki University of Technology, Laboratory of Computer and Information Science.

Kung, S. Y. (1993). Digital neural networks. New Jersey: PTR Prentice Hall.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1999). Gradient-based learning applied to document recognition, Proceedings of the IEEE, 1999

McLachlan, G. J., & Basford, K. E. (1988). Mixture models. Inference and applications to clustering. New York: Marcel Dekker

Murphy, P. M., & Aha, D. W. (1994). UCI Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, Department of Information and Computer Science.

Oehler, K.L, & Gray, R.M. (1995). Combining image compression and classification using vector quantization. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17, 461-473.

Oja, E. (1983). Subspace methods for pattern recognition. Letchworth: Research Studies Press.

Perlmutter, K.O., Perlmutter, S. M., Gray, R.M., Olshen, R.A., & Oehler, K.L. (1996). Bayes risk weighted vector quantization with posterior estimation for image compresion and classification. IEEE Transactions on Image Processing, 5, 347-360.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., & Flannery, B.P. (1992). Numerical recipes in C: The art of scientific computing. 2nd Edition. Cambridge: Cambridge University Press

Sckölkopf, B., Smola, A., & Müller, K. (1998). Non-linear component analysis as a kernel eigenvaule problem. Neural Computation, 10, 1299-1319.

Sirosh, J. (1995). A self-organising neural network model of the primary visual cortex, PhD. Thesis. Austin, TX: The University of Texas at Austin.

Thrun, S., & Pratt, L. (1998). Learning to Learn. Boston, MA: Kluwer Academic Publishers.

Wold, S. (1976). Pattern recognition by means of disjoint principal components models. Pattern Recognition, 8, 127-139.