PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 44, NO. 1, PP. 77-90 (2000)

PHONEME CLASSIFICATION USING KERNEL PRINCIPAL COMPONENT ANALYSIS

András KOCSOR[1], András KUBA[2] and László TÓTH[3]

Research Group on Artificial Intelligence
of the Hungarian Academy of Sciences
and of the University of Szeged,
H-6720 Szeged, Aradi vértanúk tere 1., Hungary
e-mail: { [1]kocsor, [2]andkuba, [3]tothl }@inf.u-szeged.hu
URL: http://www.inf.u-szeged.hu/speech

Received: Aug. 31, 2001

Abstract

A substantial number of linear and nonlinear feature space transformation methods have been proposed in recent years. Using the so-called 'kernel-idea' well-known linear techniques such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Independent Component Analysis (ICA) can be non-linearized in a general way. The aim of this paper here is twofold. First, we describe this general non-linearization technique for linear feature space transformation methods. Second, we derive formulas for the ubiquitous PCA technique and its kernel version, first proposed by SCHÖLKOPF et al., using this general schema and we examine how this transformation affects the efficiency of several learning algorithms applied to the phoneme classification task.

Keywords: kernel methods, feature space transformation, Principal Component Analysis

1. Introduction

In an earlier paper [7] we compared the effects of the linear feature space transformation methods Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Independent Component Analysis (ICA) on a number of learning algorithms. The algorithms compared were TiMBL (the IB1 algorithm), C4.5 (ID3 tree learning), OC1 (oblique tree learning), Artificial Neural Nets (ANN), Gaussian Mixture Modelling (GMM) and Hidden Markov Modelling (HMM). The domain of the comparison was phoneme classification using a certain segmental phoneme model, and each learner was tested with each transformation in order to find the best combination. In addition, in that paper we experimented with several feature sets such as filter bank energies, mel-frequency cepstral coefficients (MFCC) and gravity centers. This paper extends our investigations into nonlinear methods. Namely, similar to SCHÖLKOPF et al. [9] but in a different way we show how the well-known Principal Component Analysis (PCA) can be non-linearized using the so-called 'kernel-idea'. Besides presenting the 'kernel-idea' we also give formulas both for the original PCA and the kernel-PCA. In this paper we systematically examine how this nonlinear feature transformation affects the efficiency of several learning algorithms. As mentioned previously in our earlier study we experimented

with several feature sets and we found the best one to be the critical band log-energies and its derivatives. So now we only use this quite traditional technique to extract frame-based features from the speech signal. We also learned from our previous investigations that from the learning algorithms PCA [7] was the most suitable for GMM [1] and ANN [1]. Thus, in this paper we present classification results only for these two methods completed with the general purpose Support Vector Machine (SVM) and a HMM recognizer.

The structure of the paper is as follows. First, we discuss linear feature space transformation methods and afterwards describe the way that can be used to obtain the kernel counterpart. Then we examine the technical points of the PCA derivation, followed by the derivation of kernel-PCA along the same line. The next section presents the applied learning algorithms which is followed by the experimental results. Then we round off this paper with a discussion of conclusions and further remarks.

2. Linear Feature Space Transformation Methods with Kernels

Before executing a learning algorithm additional vector space transformations can be applied on the features obtained. The role of using these methods is twofold. Firstly they may aid classification performance, and secondly they may also reduce the dimensionality of the data. This is due to the fact that these techniques generally search for a transformation which emphasizes certain important features and suppresses or even eliminates less desirable ones.

Without loss of generality it will be assumed that the original data set lies in $\mathbb{R}^n$, and that there are s elements $\mathbf{x}_1, \ldots, \mathbf{x}_s$ in the training set and t elements $\mathbf{y}_1, \ldots, \mathbf{y}_t$ in the testing set. Any feature space transformation algorithm uses the training vectors as its input and forms a mapping $\Omega : \mathbb{R}^n \to \mathbb{R}^m$ as its output where in most cases besides the mapping the dimension reduction (represented by m) is also determined by the algorithm itself. After applying the mapping Ω the transformed training and testing vectors are denoted by $\mathbf{x}'_1, \ldots, \mathbf{x}'_s$ and $\mathbf{y}'_1, \ldots, \mathbf{y}'_t$, respectively.

2.1. Linear Feature Space Transformation Methods

With linear feature space transformation methods we search for an optimal (in some cases orthogonal) linear transformation $\mathbb{R}^n \to \mathbb{R}^m \quad (m \leq n)$ of the form $\mathbf{x}'_i = \mathbf{A}^\top \mathbf{x}_i, \ i \in \{1, \ldots, s\}$, noting that the precise definition of optimality can vary from method to method. The column vectors $\mathbf{a}_1, \ldots, \mathbf{a}_m$ of the $n \times m$ matrix $\mathbf{A}$ are assumed to be normalized.

Most of these algorithms can use an objective function $\tau() : \mathbb{R}^n \to \mathbb{R}$ which serves as a measure for selecting one optimal direction (i.e. a new base vector). Although in many cases the optimal transformation originally was defined by a

function that measures the optimality of all the m directions *together*, in most cases it is possible to define the directions of the optimal transformations one after the other by employing the τ measure for each direction separately. One quite heuristic approach is to look for unit vectors which form the stationary points of $\tau()$. Intuitively, one might think if larger values of $\tau()$ indicate better directions and the chosen directions need to be independent in certain ways, choosing stationary points that have large values is a reasonable strategy. In spite of the fact that getting these points is difficult in some cases (PCA included) we can normally find an easier way of obtaining them using eigenanalysis.

2.2. *Linear Feature Space Transformation Methods with Kernels*

In this subsection the symbols $\mathcal{H}$ and $\mathcal{F}$ denote real vector spaces that could be finite or infinite in dimension. We suppose a mapping $\Phi : \mathbb{R}^n \to \mathcal{H}$, which is not necessarily linear, and a $\dim(\mathcal{H})$ that is either finite or infinite. Furthermore, let us assume there is a *linear feature space transformation algorithm* $\mathcal{P}$ with its input formed by training points $\mathbf{x_1}, \ldots, \mathbf{x_s}$ of the vector space $\mathbb{R}^n$. We recall that the output of the algorithm is a linear transformation $\mathbb{R}^n \to \mathbb{R}^m$, where both the degree of the dimension reduction (represented by m) and the $n \times m$ transformation matrix $\mathbf{A}$ are determined by the algorithm itself. We will denote the transformation matrix $\mathbf{A}$ which results from the training data by $\mathcal{P}(\mathbf{x_1}, \ldots, \mathbf{x_s})$.

How can we obtain a non-linear feature space transformation method from $\mathcal{P}$?

First, we need to transform the training vectors into a point set in $\mathcal{H}$ by a mapping Φ and the algorithm $\mathcal{P}$ is applied on these transformed points in $\mathcal{H}$ instead of the original ones in $\mathbb{R}^n$. In this way employing the algorithm $\mathcal{P}$ on the input elements $\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_s}) \in \mathcal{H}$ we can obtain a linear transformation $\Psi : \mathcal{H} \to \mathcal{F}$. Since Φ is in general non-linear, the composite transformation $\Psi \circ \Phi$ of Φ and Ψ will not necessarily be linear either. Much like the above let us denote the matrix of the resulting linear mapping Ψ by $\mathcal{P}(\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_s}))$. The complexity of the linear methods is usually a non-linear function of the dimensionality of the input vectors and s. Thus if $\dim(\mathcal{H})$ is much larger than n, the corresponding $\mathcal{P}$ algorithms in $\mathcal{H}$ may become unfeasible practically, so we need to replace the algorithm $\mathcal{P}$ by a new more suitable one (denote it by $\mathcal{P}'$).

How can we replace $\mathcal{P}$ by $\mathcal{P}'$?

The algorithm $\mathcal{P}$ is replaced by an equivalent algorithm $\mathcal{P}'$ for which the following property holds:

$$\mathcal{P}(\mathbf{x_1}, \ldots, \mathbf{x_s}) = \mathcal{P}'(\mathbf{x_1}^\top \mathbf{x_1}, \ldots, \mathbf{x_i}^\top \mathbf{x_j}, \ldots, \mathbf{x_s}^\top \mathbf{x_s}),$$

for arbitrary $\mathbf{x_1}, \ldots, \mathbf{x_s}$.
This will of course also hold in $\mathcal{H}$ too that

$$\mathcal{P}(\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_s}))$$
$$= \mathcal{P}'(\Phi(\mathbf{x_1})^\top \Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_i})^\top \Phi(\mathbf{x_j}), \ldots, \Phi(\mathbf{x_s})^\top \Phi(\mathbf{x_s})),$$

for arbitrary $\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_s})$. Hence the goal of a kernel method is to form an algorithm $\mathcal{P}'$ which is equivalent to $\mathcal{P}$ but its inputs are the dot products of the inputs of $\mathcal{P}$. Here the complexity of $\mathcal{P}'$ is usually a non-linear function of the complexity of the dot products in $\mathcal{H}$ and s.

How can we calculate dot products with low complexity using kernel functions?

If we have a low-complexity (perhaps linear) kernel function $\kappa() : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ for which $\Phi(\mathbf{x})^\top \Phi(\mathbf{y}) = \kappa(\mathbf{x}, \mathbf{y}), \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, then $\Phi(\mathbf{x_i})^\top \Phi(\mathbf{x_j})$ can also be computed with fewer operations (for example $O(n)$) even if the dimensions of $\Phi(\mathbf{x_i})$ and $\Phi(\mathbf{x_j})$ are infinite. In practice, however, we normally tackle the problem in just the opposite way: given a $\kappa() : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ functional as kernel, we look for a mapping Φ such that $\Phi(\mathbf{x})^\top \Phi(\mathbf{y}) = \kappa(\mathbf{x}, \mathbf{y}), \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. There are several good publications about the proper choice of the kernel functions, and also about their theory in general [13].

The two most popular kernel functions are the following:

$$\kappa_1(\mathbf{x}, \mathbf{y}) = \left(\mathbf{x}^\top \mathbf{y} + 1\right)^p, \quad p \in \mathbb{N}, \qquad \kappa_2(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{-||\mathbf{x} - \mathbf{y}||^2}{r}\right), \quad r \in \mathbb{R}^+.$$

So, after choosing a kernel function the only thing remaining is to take the $\mathcal{P}'$ version of the original algorithm (here the PCA), and replace the input elements $\mathbf{x_1}^\top \mathbf{x_1}$, $\ldots$, $\mathbf{x_i}^\top \mathbf{x_j}$, $\ldots$, $\mathbf{x_s}^\top \mathbf{x_s}$ with the elements $\kappa(\mathbf{x_1}, \mathbf{x_1})$, $\ldots$, $\kappa(\mathbf{x_i}, \mathbf{x_j})$, $\ldots$, $\kappa(\mathbf{x_s}, \mathbf{x_s})$. The algorithm that results from this substitution can carry out the transformations with a practically acceptable complexity even in infinite dimensional spaces. This transformation (together with a properly chosen kernel function) results in a non-linear feature space transformation. In the following sections we briefly describe the original (linear) PCA method and afterwards present its kernel analogues via the transformation $\mathcal{P} \to \mathcal{P}'$.

3. Principal Component Analysis

In this section a discussion of the PCA [5][2][12] and kernel-PCA [9] methods will be divided into three steps:

Preprocessing Step Describes the preprocessing that might be required by the method.

Transformation Step Here we derive the algorithms themselves.
Transformation of Test Vectors Here we discuss after having obtained a transformation based on the training vectors, what kind of processing need to be applied on the test vectors.

3.1. Principal Component Analysis

As PCA behaves very sensitively when the magnitude of the components in the feature vector is significantly different, some preprocessing steps need to be performed. First the data is standardized, the mean vector of the training data is the zero vector and the deviance of each component is 1.

Preprocessing Step:

I. Centering: We shift the original sample set $\mathbf{x_1}, \ldots, \mathbf{x_s}$ with its mean $\boldsymbol{\mu}$, to obtain a set $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_s$, with a mean of $\mathbf{0}$:

$$\tilde{\mathbf{x}}_1 = \mathbf{x}_1 - \boldsymbol{\mu}, \ldots, \tilde{\mathbf{x}}_s = \mathbf{x}_s - \boldsymbol{\mu}, \qquad \boldsymbol{\mu} = \frac{1}{s}\sum_{i=1}^{s} \mathbf{x}_i. \tag{1}$$

II. Deviance Normalization[1] *:* We multiply each component of the centered data vectors $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_s$ by the deviance of the component:

$$\hat{\mathbf{x}}_1 = \begin{pmatrix} \tilde{\mathbf{x}}_1^\top \mathbf{e}_1/\sigma_1 \\ \vdots \\ \tilde{\mathbf{x}}_1^\top \mathbf{e}_s/\sigma_s \end{pmatrix}, \ldots, \hat{\mathbf{x}}_s = \begin{pmatrix} \tilde{\mathbf{x}}_s^\top \mathbf{e}_1/\sigma_1 \\ \vdots \\ \tilde{\mathbf{x}}_s^\top \mathbf{e}_s/\sigma_s \end{pmatrix}, \tag{2}$$

where

$$\sigma_i = \left(\sum_{j=1}^{s} \left(\tilde{\mathbf{x}}_j^\top \mathbf{e}_i\right)^2\right)^{1/2}, \quad i \in \{1, \ldots, s\}$$

and $\mathbf{e_i}$ is the ith unit vector.

Transformation Step:

Normally in PCA

$$\tau(\mathbf{a}) = \frac{\mathbf{a}^\top \mathbf{C}\mathbf{a}}{\mathbf{a}^\top \mathbf{a}}, \qquad \mathbf{a} \in \mathbb{R}^n \setminus \{\mathbf{0}\}, \tag{3}$$

[1]This step may be omitted as well.

where $\mathbf{C}$ is the sample covariance matrix for the standardized data:

$$\mathbf{C} = \frac{1}{s} \sum_{i=1}^{s} \hat{\mathbf{x}}_i \hat{\mathbf{x}}_i^\top. \tag{4}$$

Practically speaking, (3) defines $\tau(\mathbf{a})$ as the variance of the $\{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_s\}$ n-dimensional point-set projected onto the vector $\mathbf{a}$. Therefore this method prefers directions having a large variance. It can be shown that stationary points of (3) correspond to the right eigenvectors of the sample covariance matrix $\mathbf{C}$ where the eigenvalues form the corresponding function values. Thus it is worth defining PCA based on the stationary points where the function $\tau()$ has dominant values. If we assume that the eigenpairs of $\mathbf{C}$ are $(\mathbf{c}_1, \lambda_1), \ldots, (\mathbf{c}_n, \lambda_n)$ and $\lambda_1 \geq \ldots \geq \lambda_n$, then the transformation matrix $\mathbf{A}$ will be $[\mathbf{c}_1, \ldots, \mathbf{c}_m]$, i.e. the eigenvectors with the largest m eigenvalues. Since the sample covariance matrix $\mathbf{C}$ is a symmetric positive semidefinite, the eigenvectors are orthogonal and the corresponding real eigenvalues are nonnegative. After this orthogonal linear transformation the dimensionality of the data will be m. It is easy to check that the sample $\mathbf{x}'_i = \mathbf{A}^\top \hat{\mathbf{x}}_i, \quad i \in \{1, \ldots, s\}$ represented in the new orthogonal basis will be uncorrelated, i.e. the covariance matrix $\mathbf{C}'$ of $\mathbf{x}'$ is diagonal. The diagonal elements of $\mathbf{C}'$ are the m dominant eigenvalues of $\mathbf{C}$.

In our experiments, m (the dimensionality of the transformed space) was chosen to be the smallest integer for which

$$\frac{\lambda_1 + \ldots + \lambda_m}{\lambda_1 + \ldots + \lambda_n} > 0.99 \tag{5}$$

holds. Note that there are many other alternatives for finding a reasonable value of m.

Transformation of Test Vectors:

For an arbitrary test vector $\mathbf{y}$:

$$\mathbf{y}' = \mathbf{A}^\top \hat{\mathbf{y}},$$

where $\hat{\mathbf{y}}$ denotes the preprocessed $\mathbf{y}$.

3.2. *Formulas for Kernel-PCA*

Having chosen a proper κ kernel function for which

$$\kappa(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^\top \Phi(\mathbf{y}), \qquad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n,$$

holds for a mapping $\Phi : \mathbb{R}^n \to \mathcal{H}$, we now give the PCA transformation in $\mathcal{H}$.

Preprocessing Step:[2]

I. Kernel Centering: We shift the data $\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_s})$ with its mean $\boldsymbol{\mu}^{\Phi}$, to obtain a set $\hat{\Phi}(\mathbf{x_1}), \ldots, \hat{\Phi}(\mathbf{x_s})$ with a mean of $\mathbf{0}$:

$$\hat{\Phi}(\mathbf{x_1}) = \Phi(\mathbf{x_1}) - \boldsymbol{\mu}^{\Phi}, \ldots, \hat{\Phi}(\mathbf{x_s}) = \Phi(\mathbf{x_s}) - \boldsymbol{\mu}^{\Phi}, \quad \boldsymbol{\mu}^{\Phi} = \frac{1}{s}\sum_{i=1}^{s} \Phi(\mathbf{x_i}). \tag{6}$$

Transformation Step:

We employed the following metric in $\mathcal{H}$:

$$\tau^{\hat{\Phi}}(\mathbf{a}) = \frac{\mathbf{a}^{\top}\mathbf{C}^{\hat{\Phi}}\mathbf{a}}{\mathbf{a}^{\top}\mathbf{a}}, \qquad \mathbf{a} \in \mathcal{H} \setminus \{\mathbf{0}\}, \tag{7}$$

where $\mathbf{C}^{\hat{\Phi}}$ is the covariance matrix of the sample $\hat{\Phi}(\mathbf{x_1}), \ldots, \hat{\Phi}(\mathbf{x_s})$:

$$\mathbf{C}^{\hat{\Phi}} = \frac{1}{s}\sum_{i=1}^{s} \hat{\Phi}(\mathbf{x_i})\hat{\Phi}(\mathbf{x_i})^{\top}. \tag{8}$$

Much like the PCA approach we define the kernel-PCA based on the stationary points of (7) which are given as the eigenvectors of the symmetric positive semidefinite matrix $\mathbf{C}^{\hat{\Phi}}$. Because of the special form of $\mathbf{C}^{\hat{\Phi}}$ we can suppose the following equation to hold during the study of the stationary points: [3]

$$\mathbf{a} = \sum_{i=1}^{s} \alpha_i \hat{\Phi}(x_i). \tag{9}$$

The following formulas give $\tau^{\hat{\Phi}}(\mathbf{a})$ as the function of $\boldsymbol{\alpha}$ and $\kappa(\mathbf{x_i}, \mathbf{x_j})$

$$\tau^{\hat{\Phi}}(\mathbf{a}) = \frac{\mathbf{a}^{\top}\mathbf{C}^{\hat{\Phi}}\mathbf{a}}{\mathbf{a}^{\top}\mathbf{a}} = \frac{\left(\sum_{t=1}^{s} \alpha_t \hat{\Phi}(x_t)^{\top}\right)\mathbf{C}^{\hat{\Phi}}\left(\sum_{k=1}^{s} \alpha_k \hat{\Phi}(x_k)\right)}{\left(\sum_{t=1}^{s} \alpha_t \hat{\Phi}(x_t)^{\top}\right)\left(\sum_{k=1}^{s} \alpha_k \hat{\Phi}(x_k)\right)} = \frac{\boldsymbol{\alpha}^{\top}\frac{1}{s}\mathbf{K}^{\hat{\Phi}}\mathbf{K}^{\hat{\Phi}}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^{\top}\mathbf{K}^{\hat{\Phi}}\boldsymbol{\alpha}}, \tag{10}$$

[2] We have to mention here that the 'Deviance Normalization' we have seen at PCA is practically impossible because, in the general case, we do not know the components of the vectors $\Phi(\mathbf{x})$in $\mathcal{H}$.

[3] We can arrive at this assumption in many other ways, e.g. we can decompose an arbitrary vector $\mathbf{a}$ as $\mathbf{a_1} + \mathbf{a_2}$, where $\mathbf{a_1}$ gives that component of $\mathbf{a}$ which falls in $SPAN(\hat{\Phi}(\mathbf{x_1}), \ldots, \hat{\Phi}(\mathbf{x_s}))$, while $\mathbf{a_2}$ gives the component perpendicular to it. Then from the derivation of (7) we see that $\mathbf{a_2}^{\top}\mathbf{a_2} = 0$ for the stationary points.

where[4]

$$\mathbf{K}_{ik}^{\hat{\Phi}} = \left(\Phi(\mathbf{x_t})^\top - \left(\tfrac{1}{s}\sum_{i=1}^{s}\Phi(\mathbf{x_i})^\top\right)\right)\left(\Phi(\mathbf{x_k}) - \left(\tfrac{1}{s}\sum_{i=1}^{s}\Phi(\mathbf{x_i})\right)\right) = \kappa(\mathbf{x_t},\mathbf{x_k}) - \left(\tfrac{1}{s}\sum_{i=1}^{s}(\kappa(\mathbf{x_i},\mathbf{x_k}) + \kappa(\mathbf{x_t},\mathbf{x_i}))\right) + \tfrac{1}{s^2}\sum_{i=1}^{s}\sum_{j=1}^{s}\kappa(\mathbf{x_i},\mathbf{x_j}). \tag{11}$$

From differentiating (10) with respect to $\boldsymbol{\alpha}$ we get that the stationary points are the solution vectors of the general eigenvalue problem $\frac{1}{s}\mathbf{K}^{\hat{\Phi}}\mathbf{K}^{\hat{\Phi}}\boldsymbol{\alpha} = \lambda\mathbf{K}^{\hat{\Phi}}\boldsymbol{\alpha}$, which in this case is obviously equivalent to the problem $\frac{1}{s}\mathbf{K}^{\hat{\Phi}}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$. Furthermore, since $\kappa(\mathbf{x_t},\mathbf{x_k}) = \kappa(\mathbf{x_k},\mathbf{x_t})$ and[5] $\boldsymbol{\alpha}^\top\frac{1}{s}\mathbf{K}^{\hat{\Phi}}\boldsymbol{\alpha} = \frac{1}{s}\mathbf{a}^\top\mathbf{a} \geq 0$, the matrix $\frac{1}{s}\mathbf{K}^{\hat{\Phi}}$ is a symmetric positive semidefinite and thus its eigenvectors are orthogonal and the corresponding real eigenvalues are non-negative. Let the m positive dominant eigenvalues of $\frac{1}{s}\mathbf{K}^{\hat{\Phi}}$ be denoted by $\lambda_1 \geq \ldots \geq \lambda_m > 0$ and the corresponding normalized eigenvectors be $\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^m$. Then the orthogonal matrix of the transformation we need can be calculated as below.

$$\mathbf{A}_{\hat{\Phi}} := \left[\frac{1}{\sqrt{s\lambda_1}}\sum_{i=1}^{s}\alpha_i^1\hat{\Phi}(\mathbf{x_i}), \ldots, \frac{1}{\sqrt{s\lambda_m}}\sum_{i=1}^{s}\alpha_i^m\hat{\Phi}(\mathbf{x_i})\right], \tag{12}$$

where the factors $1/\sqrt{s\lambda}$ are needed to keep the column vectors of $\mathbf{A}_{\hat{\Phi}}$ normalized.

Transformation of Test Vectors:

Let $\mathbf{y}$ be an arbitrary test vector. After preprocessing $\Phi(\mathbf{y})$ we find that $\hat{\Phi}(\mathbf{y}) = \Phi(\mathbf{y}) - \boldsymbol{\mu}^{\Phi}$. Then

$$\mathbf{y}' = \mathbf{A}_{\hat{\Phi}}^\top\hat{\Phi}(\mathbf{y}) = \left[\frac{1}{\sqrt{s\lambda_1}}\sum_{i=1}^{s}\alpha_i^1 c_i, \ldots, \frac{1}{\sqrt{s\lambda_m}}\sum_{i=1}^{s}\alpha_i^m c_i\right]^\top, \tag{13}$$

where

$$c_i = \hat{\Phi}(\mathbf{x_i})^\top\hat{\Phi}(\mathbf{y}) = \kappa(\mathbf{x_i},\mathbf{y}) - \left(\frac{1}{s}\sum_{j=1}^{s}\left(\kappa(\mathbf{x_i},\mathbf{x_j}) + \kappa(\mathbf{x_j},\mathbf{y})\right)\right) + \frac{1}{s^2}\sum_{t=1}^{s}\sum_{j=1}^{s}\kappa(\mathbf{x_t},\mathbf{x_j}). \tag{14}$$

In our experience the strategy for obtaining a suitable m was identical to that in PCA (5).

[4] SCHÖLKOPF et al. give $\mathbf{K}^{\hat{\Phi}}$ in a matrix form using additional matrices. Our formula, however, turned out to be easier to code, and resulted in a more effective program.

[5] Here we temporarily disregard the constraint $\mathbf{a} \neq 0$.

3.3. Summary

Stated briefly the most important properties of the techniques discussed above are:

> *PCA* concentrates on those independent directions with the largest variances. *Kernel-PCA* is a non-linearized version of PCA where the non-linearization was carried out using the 'kernel idea'. In this method the original algorithm is executed in a transformed (probably infinite) feature space $\mathcal{H}$; the kernel function κ gives implicit access to the elements of this space.

4. Experimental Results

The automatic speech recognition is a special pattern classification problem where one of the dimensions of the pattern is time. Speech signals show very specific dynamic variations along this axis, and thus require dedicated recognition techniques. One approach is to segment the speech signal into its supposed building blocks (e.g. phonemes), recognize these separately and then combine the recognition scores for the whole signal. Because of the difficulties of segmentation, however, hidden Markov modelling (HMM) became the dominant technology instead, in which utterances are processed in small uniform chunks (called frames), and their time variability is handled via a neat probabilistic structure. Lately HMM has received a lot of bad press over its time modelling capabilities, and there have been efforts towards generalizations which work with phonetic segments rather than frames. We restrict our investigations here only to the phoneme classification task. Word level results and a description of the technique beyond the phoneme level are dealt with in a separate report.

4.1. Evaluation Domain

The feature space transformation and the classification techniques were compared using a relatively small[6] corpus which consists of several speakers pronouncing Hungarian numbers. More precisely, 20 speakers were used for training and 6 for testing, and 52 utterances were recorded from each person. The ratio of male and female talkers was 50%-50% in both the training and testing sets. The recordings were made using a cheap commercial microphone in a reasonably quiet environment, at a sample rate of 22050 Hz. The whole corpus was manually segmented and labeled. Since the corpus contained only numbers, we had occurrences of only 32 phones, which is approximately two thirds of the Hungarian phoneme set. Since some of these labels represented only allophonic variations of the same phoneme

[6] Our reason for employing such a limited database was that we chose to work with Hungarian and no larger (segmented) corpus of Hungarian was available at the time of writing. However, one of our aims for the future will be to conduct additional tests on a larger database.

some labels were fused and so we actually only worked with a set of 28 labels. The number of occurrences of the different labels in the training set was between 40 and 599.

4.2. Segmental Features

In the following we describe the feature extraction techniques which were used in our tests.[7] Although there are many sophisticated segmental models offered in the literature (e.g. [3]), we employed a simple technique similar to that of the SUMMIT system [4]. At the frame level the speech signals were represented by their critical-band log-energies, and the averages of the 24 critical-band log-energies[8] of the segment thirds (divided in a 1-2-1 ratio) were used as segmental features for phoneme classification. The advantage of this method is that it needs only trifling additional calculations following the computation of the frame-based features. Moreover, it returns the same number of segmental features independent of the segment length, which was a prerequisite for the classifiers used.

We also made use of the variances of the features along the segments to filter out candidates that contain boundaries inside them, and the derivatives of the features at the boundaries to remove candidates with improbable start and end-points. These segmental features were calculated only on 4 wide frequency bands as this proved quite sufficient.

A special segmental feature is the duration of the phoneme. We consider it especially important for languages like Hungarian where phonemic duration can play a discriminative role. As our preliminary experiments found duration to be very useful indeed, it was employed as a segmental feature in all our experiments. Thus, including duration, 77 features were used altogether to represent the segments.

4.3. Learning Methods

The statistical learning methods employed in classification problems are called either discriminative or generative, depending on what they model. Discriminative models describe the common feature space of all the classes, and focus on discriminating one class from another. They do this either by finding proper parameters for a set of separating surfaces of a given type (parametric modelling), or by representing the classes with elements and distance metrics (non-parametric modelling).

[7] The only exception was the HMM recognizer, which had its own features (see sec. 4.3 for details).

[8] The signals were processed in 512-point frames (23.2 ms), where the frames overlapped by a factor of 3/4. A Fast Fourier transform was applied on the frames. After that critical band energies were approximated by the use of triangular-shaped weighting functions. 24 such filters were used to cover the frequency range from 0 to 11025 Hz, the bandwidth of each filter being 1 bark. The energy values were then measured on a logarithmic scale.

In our paper artificial neural networks (ANN) and a support vector machine (SVM) represent the class of discriminative learners.

According to Bayes' law, the conditional probability $P(C|\mathbf{x})$ of a class C for a vector $\mathbf{x}$ can be obtained from the formula $P(C|\mathbf{x}) = P(\mathbf{x}|C)P(C)/P(\mathbf{x})$. Thus, instead of modelling $P(C|\mathbf{x})$ directly as discriminative models do, another possibility is to estimate the class-conditional probabilities $P(\mathbf{x}|C)$ for each class separately. This is the so-called generative modelling approach. And though it may seem a disadvantage that a priori probabilities $P(C)$ also have to be estimated, this decomposition is actually very useful in speech recognition as 'acoustic' and 'language' models can then be handled separately. From the techniques studied in our paper HMM and Gaussian mixture models (GMM) belong to the class of generative learners.

Hidden Markov modelling (HMM) is currently the dominant technology in speech recognition [8]. This is why in the tests the HMM was trained on its 'standard' features and not on those used in all the other experiments. The intention behind this was to have a reference point for the current state-of-the-art technology to judge things by. The hidden Markov models for the HMM experiments were trained using the FlexiVoice speech engine [11]. The system used a feature vector of 34 components, which consisted of 16 LPC-derived cepstrum coefficients plus the frame energy, and the first derivatives of these. The frame size was 30 ms while the step size was 10 ms. One model was assigned to each of the phonemes. The phoneme models were of the three-state strictly left-to-right type, that is each state had one self transition and one transition to the next state. In each case the observations were modelled using a mixture of four Gaussians with diagonal covariance matrices. The models were trained using the Viterbi training algorithm.

Gaussian mixture modelling assumes that the class-conditional probability distribution $P(\mathbf{x}|C)$ can be well approximated by a distribution of the form

$$f(\mathbf{x}) = \sum_{i=1}^{k} c_i \mathcal{N}(\mathbf{x}, \mu_i, \mathbf{C_i}), \tag{15}$$

where $\mathcal{N}(\mathbf{x}, \mu_i, \mathbf{C_i})$ denotes the multidimensional normal distribution with mean μ_i and covariance matrix $\mathbf{C_i}$, k is the number of mixtures, and c_i are non-negative weighting factors which sum to 1.

Unfortunately, there is no closed formula for the optimal parameters of the mixture model, so normally the expectation-maximization (EM) algorithm is used to find proper parameters, but it guarantees only a locally optimal solution. This iterative technique is very sensitive to initial parameter values, so we used k-means clustering [8] to find a good starting parameter set. Since k-means clustering again guaranteed finding only a local optimum, we ran it 15 times with random parameters and used the one with the highest log-likelihood to initialize the EM algorithm. After experimenting the best value for the number of mixtures k was found to be 3. In all cases the covariance matrices were forced to be diagonal as training full matrices would have required much more training data (and computation time as well).

Artificial Neural Networks (ANN) [10] count now among the conventional pattern recognition tools, so we will not describe them here. In the experiments we used the most common feed-forward multilayer perceptron network with the backpropagation learning rule. The number of neurons in the hidden layer was set to be three times the number of features (this value was chosen empirically based on preliminary experiments). The training was stopped when, for the last 20 iterations, the decrease in the error between two consecutive iteration steps stayed below a given threshold.

Support Vector Machine (SVM) With the classification task we also conducted tests with a promising new technique called Support Vector Machine. Rather than briefly describing this method for an overview we refer the interested reader to [13]. In all experiments with SVM a second-order polynomial kernel function was applied.

4.4. Evaluation Method

The learning methods which model the a posteriori probabilities $P(C|\mathbf{x})$ return a probability value for each class C given a test vector $\mathbf{x}$. The so-called Bayes's decision rule states [10] that the optimal way of converting these values to a class label is to choose the class with the maximum a posteriori probability. We used this rule to calculate the classification error for these techniques.

Finally, some of the learning techniques (HMM, GMM) model the class-conditional probabilities $P(\mathbf{x}|C)$. From this $P(C|\mathbf{x})$ can be obtained by employing the Bayes decision rule and we have to choose the class for which $P(\mathbf{x}|C)P(C)$ is maximal. ($P(\mathbf{x})$ is independent of C and so can be omitted.) Instead of doing this we did not multiply by the factor $P(C)$ in the evaluation, since handling this probability traditionally belongs to the language model. Also, preliminary tests showed that multiplication with $P(C)$ led only to marginal improvements, clearly because the relative frequencies of the phonemes were quite well balanced.

4.5. Experiments

The experiments were performed on seven feature sets. One of them was the original one containing the addressed 77 features, the others being the transformed versions using PCA and kernel-PCA with various kernel functions. In the experiments we used 5616 training and 1692 test examples. Throughout the kernel-PCA procedure we computed the matrix $\mathbf{K}^{\Phi}$ from all training examples and polynomial kernels with various exponents were used.[9] *Table 1* shows the recognition accuracies where the

[9]Each kernel-PCA algorithm ran for about 8 cpu hours using an Intel PII-350Mhz computer with a 512Mb operative memory.

Table 1 Recognition accuracies for the phoneme classification. For comparison, HMM scored 90.66% . The maximum is typeset in bold.

	none	PCA $(\mathbf{x}^\top\mathbf{y})$	KPCA $(\mathbf{x}^\top\mathbf{y})^{1.005}$	KPCA $(\mathbf{x}^\top\mathbf{y})^{1.01}$	KPCA $(\mathbf{x}^\top\mathbf{y})^{1.05}$	KPCA $(\mathbf{x}^\top\mathbf{y})^{1.1}$	KPCA $(\mathbf{x}^\top\mathbf{y})^{1.5}$
ANN	93.38	94.09	94.39	94.42	94.27	94.15	92.61
SVM	94.11	94.24	94.75	**94.83**	94.22	93.71	87.83
GMM	80.08	88.32	89.63	89.94	88.71	87.12	79.73

columns represent the seven feature sets (transformed/not-transformed) while the rows correspond to the applied learning methods.

4.6. Discussion

When inspecting the results the first thing one notices is that the segmental discriminative models[10](ANN, SVM) significantly outperformed the results of the HMM and the segmental generative model[11](GMM). Considering ANN, GMM and SVM, the average classification results were 93.9%, 93.4% and 86.2%, respectively. SVM reached the best recognition accuracy which was 94.8%. On examining the effect of the PCA and kernel-PCA we found that kernel-PCAs with an exponent near to 1(weak non-linearity) are more suitable to increase the recognition accuracies than the strongly non-linear ones.[12] Another thing we realized was that the efficiency of the recognition was mostly improved by the PCA and kernel-PCA in the case of GMM which is due to the diagonal form of the covariance matrix which was used to approximate the multidimensional normal distribution.

Finally, we mention that the conclusions above were drawn from visual inspection of the results. For a rigorous justification of our impressions we also ran significance tests. More precisely, paired two-sided t-tests were applied at the 5% significance level. While these tests confirmed that ANN and SVM were equally the best learning techniques then considering the feature space transformation methods kernel-PCA with the kernel function $(\mathbf{x}^\top\mathbf{y})^{1.01}$ proved to be the best one.

5. Conclusions

As regards the PCA with kernels, we have to conclude that in general it is worth going on with experimenting this type of non-linearity. Our experiments obviously

[10]segmental model + discriminative learners

[11]segmental model + generative learners

[12]We also did experiments with other kernels using various normalization methods, but we found that strong non-linearity was unsuitable for the classification technique.

show that kernel-PCAs with a weak non-linearity combined with ANN or SVM are competitive or even superior to the other state-of-the-art algorithms tested in this work. The described phoneme classification techniques can be well utilized by a segmental speech recognizer. Future research will focus on incorporating these techniques into our speech recognizer. Above all, we plan to do experiments with a kernelized version of Independent Component Analysis and Linear Discriminant Analysis. In this paper similar to [7] we have sought the best combination of certain classification methods and feature space transformation methods (i.e. the PCA and kernel-PCA with various parameters). Instead of selecting the most suitable combination in the future we plan to use or develop classification methods which implicitly imply an optimal non-linear feature space transformation.

Acknowledgments

The HMM system used in our experiments [11] was trained and tested by Máté Szarvas at the Department of Telecommunications and Telematics, Technical University of Budapest. We greatly appreciate his help which was indispensable in making this study complete.

References

[1] ALDER, M. D., *Principles of Pattern Classification: Statistical, Neural Net and Syntactic Methods of Getting Robots to See and Hear,* http://ciips.ee.uwa.edu.au/~mike/PatRec, 1994.
[2] BATTLE, E. – NADEU, C. – FONOLLOSA, J. A. R., Feature Decorrelation Methods in Speech Recognition. A Comparative Study. In *Proceedings of ICSLP'98*, 1998.
[3] FUKADA, T. – SAGISAKA, Y. – PALIWAL, K. K., Model Parameter Estimation for Mixture Density Polynomial Segment Models, *Proc. of ICASSP'97*, pp. 1403-1406, Munich, Germany, 1997.
[4] HALBERSTADT, A. K., Heterogeneous Measurements and Multiple Classifiers for Speech Recognition, Ph.D. Thesis, Dep. Electrical Engineering and Computer Science, MIT, 1998.
[5] JOLLIFFE, I. J., *Principal Component Analysis,* Springer-Verlag, New York, 1986.
[6] KOCSOR, A. – KUBA, A. Jr. – TÓTH, L., An Overview of the OASIS Speech Recognition Project, *In Proceedings of ICAI'99*, 1999.
[7] KOCSOR, A. – TÓTH, L. – KUBA, A. Jr. – KOVÁCS, K. – JELASITY, M. – GYIMÓTHY, T. – CSIRIK, J., A Comparative Study of Several Feature Transformation and Learning Methods for Phoneme Classification, *International Journal of Speech Technology*, **3**, Numbers 3/4, pp. 263–276, 2000.
[8] RABINER, L. – JUANG, B.-H., *Fundamentals of Speech Recognition,* Prentice Hall, 1993.
[9] SCHÖLKOPF, B. – MIKA, S. – BURGES, C. J. C. – KNIRSCH, P. – MÜLLER, K. – RÄTSCH, G. – SMOLA, A. J., Input Space vs. Feature Space in Kernel-Based Methods, *IEEE Transactions on Neural Networks*, **10** (5) (1999), pp. 1000-1017.
[10] SCHÜRMANN, J., *Pattern Classification, A Unified View of Statistical and Neural Approaches,* Wiley & Sons, 1996.
[11] SZARVAS, M. – MIHAJLIK, P. – FEGYÓ, T. – TATAI, P., Automatic Recognition of Hungarian: Theory and Practice, *International Journal of Speech Technology*, **3** (2000), Numbers 3/4, pp. 237–251.
[12] TIPPING, M. E. – BISHOP, C. M., Probabilistic Principal Component Analysis, *J. R. Static. Soc. B*, **61** (1999), pp. 611–622.
[13] VAPNIK, V. N., *Statistical Learning Theory,* John Wiley & Sons Inc., 1998.