A Nonlinearized Discriminant Analysis and its Application to Speech Impediment Therapy

András Kocsor[1], László Tóth[2] and Dénes Paczolay[3]

Research Group on Artificial Intelligence
of the Hungarian Academy of Sciences
and the University of Szeged
H-6720 Szeged, Aradi vértanúk tere 1., Hungary
{[1]kocsor, [2]tothl, [3]pdenes}@inf.u-szeged.hu

Abstract. This paper studies the application of automatic phoneme classification to the computer-aided training of the speech and hearing handicapped. In particular, we focus on how efficiently discriminant analysis can reduce the number of features and increase classification performance. A nonlinear counterpart of Linear Discriminant Analysis, which is a general purpose class specific feature extractor, is presented where the nonlinearization is carried out by employing the so-called 'kernel-idea'. Then, we examine how this nonlinear extraction technique affects the efficiency of learning algorithms such as Artificial Neural Network and Support Vector Machines.

1 Speech Impediment Therapy and Real-Time Phoneme Classification

This paper deals with the application of speech recognition to the computer-aided training of the speech and hearing handicapped. The program we present was designed to help in the speech training of the hearing impaired, where the goal is to support or replace their diminished auditory feedback with a visual one. But the program could also be applied to improving the reading skills of children with reading difficulties. Experience shows that computers more readily attract the attention of young people, who are usually more willing to practice with the computer than with the traditional drills.

Since both groups of our intended users consist mostly of young children it was most important that the design of the software interface be made attractive and novel. In addition, we realized early on that the real-time visual feedback the software provides must be kept simple, otherwise the human eye cannot follow it. Basically this is why the output of a speech recognizer seems better suited to this goal than the usual method where only the short-time spectrum is displayed: a few flickering discrete symbols are much easier to follow than a spectrum curve, which requires further mental processing. This is especially the case with very young children.

From the speech recognition point of view the need for a real-time output poses a number of special problems. Owing to the need for very fast classification we cannot delay the response even until the end of phonemes, hence we cannot employ complicated long-term models. The algorithm should process no more than a few neighbouring frames. Furthermore, since the program has to recognize vowels pronounced in isolation as well, a language model cannot be applied.

In our initial experiments we focussed on the classification of vowels, as the learning of the vowels is the most challenging for the hearing-impaired. The software supposes that the vowels are pronounced in isolation or in the form of two-syllable words, which is a more usual training strategy. The program provides a visual feedback on a frame-by-frame basis in the form of flickering letters, their brightness being proportional to the speech recognizer's output (see fig.1). To see the speaker's progress over longer periods, the program can also display the recognition scores during the previous utterance (see fig.2). Of course it is always possible to examine the sample spectra as well, either on a frame-by-frame or on an utterance-based basis. The utterances can be recorded and played back for further study and analysis by the teacher.

This article describes the experiments conducted with the LDA and Kernel-LDA transforms, intended to improve and possibly speed up the classification of vowels. As for the classification itself we used neural nets (ANN) and support vector machines (SVM). The section below explains the mathematical details of the Kernel-LDA transform, which is a new non-linear extension of the traditional LDA technique[1].

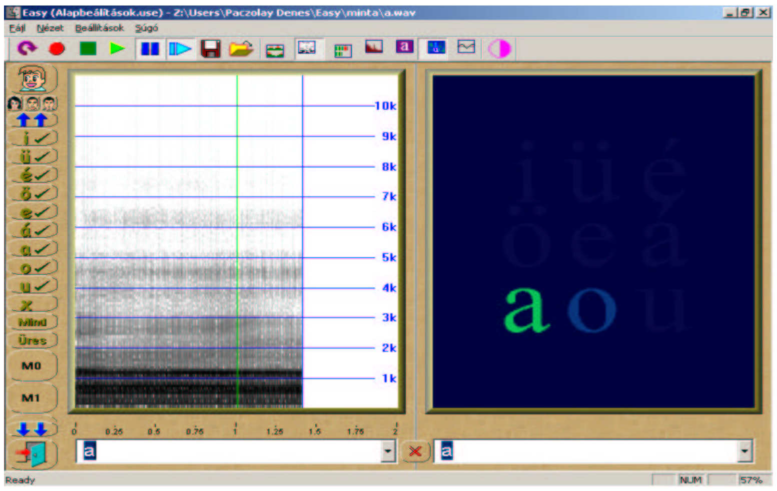
Easy (Alapbeállítások.use) - Z:\Users\Paczolay Denes\Easy\minta\a.wav
Fájl Nézet Beállítások Súgó
10k
9k
8k
7k
6k
5k
4k
3k
2k
1k
i ü é
ö e á
a o u
Mind
Üres
M0
M1
a
a
Ready
NUM
57%

Fig. 1. A screenshot from EasySpeech. The real-time response of the system for vowel /a/.

2 Linear Discriminant Analysis with and without Kernels

Before executing a learning algorithm it is a common practice to preprocess the data by extracting new features. Of the class specific feature extractors Linear Discriminant Analysis (LDA) is a traditional statistical method which has proved to be one of the

[1] In [4] this method bears the name "Kernel Fisher Discriminant Analysis". Independently of these authors we arrived to the same formulae too, the only difference being that we derived the formulae for the multiclass case, naming the technique "Kernel-LDA". Although we recently reported our results of Kernel-LDA on word recognition in [6], the method itself was not described in great detail.

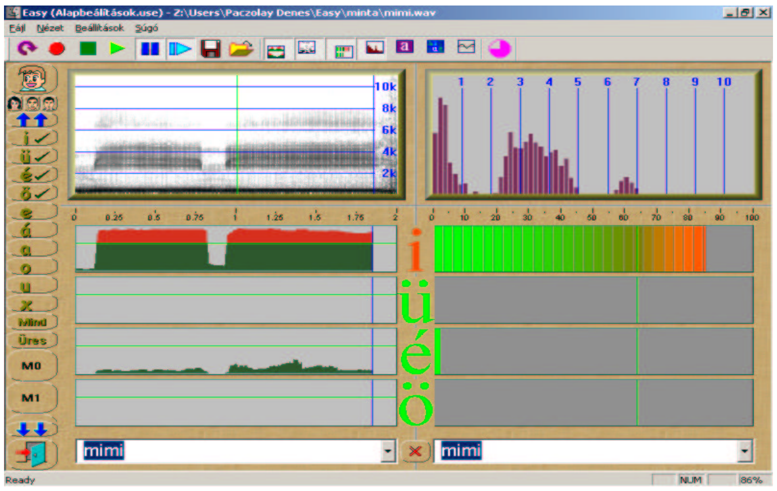
Easy (Alapbeállítások.use) - Z:\Users\Paczolay Denes\Easy\minta\mimi.wav
Fájl Nézet Beállítások Súgó
10k
8k
6k
4k
2k
1 2 3 4 5 6 7 8 9 10
i
ü
é
ö
M0
M1
mimi
mimi
Ready
NUM
86%

Fig. 2. A screenshot of EasySpeech after pronouncing the word /mimi/.

most successful preprocessing techniques in classificaton[2]. The role of this method as preprocessing is twofold. Firstly it can improve classification performance, and secondly it may also reduce the dimensionality of the data and hence significantly speed up the classification.

The goal of Linear Discriminant Analysis is to find a new (not necessarily orthogonal) basis for the data which provides the optimal separation between groups of points (classes). Without loss of generality we will assume that the original data set, i.e. the input data lies in $\mathbb{R}^n$, denoted by $\mathbf{x_1}, \ldots, \mathbf{x_r}$. The class label of each data vector is supposed to be known beforehand. Let us assume that we have k classes and an indicator function $f() : \{1, \ldots, r\} \to \{1, \ldots, k\}$, where $f(i)$ gives the class label of the point $\mathbf{x_i}$. Let r_j $(j \in \{1, \ldots, k\},\ r = r_1 + \ldots + r_k)$ denote the number of vectors associated with label j in the data. In this section we now review the formalae for LDA, and also a nonlinear extension using the so-called 'Kernel-idea'.

2.1 Linear Discriminant Analysis

In order to extract m informative features from the n-dimensional input data, we first define a function $\tau() : \mathbb{R}^n \to \mathbb{R}$ which serves as a measure for selecting the m directions (i.e. base vectors of the new basis) one at a time. For a selected direction $\mathbf{a}$ a new real valued feature can be calculated as $\mathbf{a}^\top \mathbf{x}$. Intuitively, if larger values of $\tau()$ indicate better directions and the chosen directions need to be somehow independent, choosing stationary points that have large values is a reasonable strategy. So we define a new basis for the input data based on m stationary points of τ with dominant function values. Now let us define

$$\tau(\mathbf{a}) = \frac{\mathbf{a}^\top \mathbf{B} \mathbf{a}}{\mathbf{a}^\top \mathbf{W} \mathbf{a}}, \qquad \mathbf{a} \in \mathbb{R}^n \setminus \{\mathbf{0}\}, \tag{1}$$

[2] One should note here that it can be directly used for classification as well.

where $\mathbf{B}$ is the *Between-class Scatter Matrix*, while $\mathbf{W}$ is the *Within-class Scatter Matrix*. Here *Between-class Scatter Matrix* $\mathbf{B}$ represents the scatter of the class mean vectors $\boldsymbol{\mu}_\mathbf{j}$ around the overall mean vector $\boldsymbol{\mu} = \frac{1}{r}\sum_{i=1}^{r}\mathbf{x_i}^\top\mathbf{x_i}$ while the *Within-class Scatter Matrix* $\mathbf{W}$ shows the weighted average scatter of the covariance matrices $\mathbf{C_j}$ of the sample vectors having label j:

$$\begin{array}{ll} \mathbf{B} = \sum_{j=1}^{k}\frac{r_j}{r}(\boldsymbol{\mu}_\mathbf{j} - \boldsymbol{\mu})(\boldsymbol{\mu}_\mathbf{j} - \boldsymbol{\mu})^\top & \mathbf{W} = \sum_{j=1}^{k}\frac{r_j}{r}\mathbf{C_j} \\ \mathbf{C_j} = \frac{1}{r_j}\sum_{f(i)=j}(\mathbf{x_i} - \boldsymbol{\mu}_\mathbf{j})(\mathbf{x_i} - \boldsymbol{\mu}_\mathbf{j})^\top & \boldsymbol{\mu}_\mathbf{j} = \frac{1}{r_j}\sum_{f(i)=j}\mathbf{x_i} \end{array} \tag{2}$$

Since $\tau(\mathbf{a})$ is large when its nominator is large and its denominator is small, the within-class averages of the sample projected onto $\mathbf{a}$ are far from each other, while the variance of the classes is small. The larger the value of $\tau(\mathbf{a})$ the farther the classes will be spaced and the smaller their spreads will be. It can be easily shown that stationary points of (1) correspond to the right eigenvectors of $\mathbf{W}^{-1}\mathbf{B}$, where the eigenvalues form the corresponding function values. Since $\mathbf{W}^{-1}\mathbf{B}$ is not necessarily symmetrical the number of real eigenvalues can be less than n and the corresponding eigenvectors will not necessarily be orthogonal[3]. If we select the m eigenvectors with the greatest real eigenvalues (denoted by $\mathbf{a_1}, \ldots, \mathbf{a_m}$), we will obtain new features from an arbitrary data vector $\mathbf{y} \in \mathbb{R}^n$ by $\mathbf{a_1}^\top\mathbf{y}, \ldots, \mathbf{a_m}^\top\mathbf{y}$.

2.2 Kernel-LDA

Here the symbol $\mathcal{H}$ denotes a real vector space that could be finite or infinite in dimension and we suppose a mapping $\Phi : \mathbb{R}^n \to \mathcal{H}$, which is not necessarily linear. In addition, let us assume that the algorithm of Linear Discriminant Analysis is denoted by $\mathcal{P}$ and its input is the points $\mathbf{x_1}, \ldots, \mathbf{x_r}$ of the vector space $\mathbb{R}^n$. 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 are determined by the algorithm itself. $\mathcal{P}(\mathbf{x_1}, \ldots, \mathbf{x_r})$ will denote the transformation matrix which results from the input data. Then the algorithm $\mathcal{P}$ is replaced by an equivalent algorithm $\mathcal{P}'$ for which $\mathcal{P}(\mathbf{x_1}, \ldots, \mathbf{x_r}) = \mathcal{P}'(\mathbf{x_1}^\top\mathbf{x_1}, \ldots, \mathbf{x_i}^\top\mathbf{x_j}, \ldots, \mathbf{x_r}^\top\mathbf{x_r})$ holds for arbitrary $\mathbf{x_1}, \ldots, \mathbf{x_r}$. Thus $\mathcal{P}'$ is equivalent to $\mathcal{P}$ but its inputs are the pairwised dot products of the inputs of algorithm $\mathcal{P}$. Then applying a nonlinear mapping Φ on the input data, yields a nonlinear feature transformation matrix $\mathcal{P}'(\Phi(\mathbf{x_1})^\top\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_i})^\top\Phi(\mathbf{x_j}), \ldots, \Phi(\mathbf{x_r})^\top\Phi(\mathbf{x_r}))$. These dot products can be computed here in $\mathcal{H}$ (which may be infinite in dimension), but 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 when the dimensions of $\Phi(\mathbf{x_i})$ and $\Phi(\mathbf{x_j})$ are infinite. So, after choosing a kernel function, the only thing that remains is to take the algorithm $\mathcal{P}'$ and replace the input elements $\mathbf{x_1}^\top\mathbf{x_1}$, $\ldots$, $\mathbf{x_i}^\top\mathbf{x_j}$, $\ldots$,

[3] Besides this, numerical problems can occure during the computation of $\mathbf{W}^{-1}$ if $det(\mathbf{W})$ is near zero. The most probable cause for this could be the redundancy of feature components. But we know $\mathbf{W}$ is positive semidefinite. So if we add a small positive constant ϵ to its diagonal, that is we work with $\mathbf{W} + \epsilon\mathbf{I}$ instead of $\mathbf{W}$, this matrix is guaranteed to be positive definite and hence should always be invertible. This small act of cheating can have only a negligible effect on the stationary points of (1).

$\mathbf{x_r}^\top \mathbf{x_r}$ with the elements $\kappa(\mathbf{x_1},\mathbf{x_1}),\ldots,\kappa(\mathbf{x_i},\mathbf{x_j}),\ldots,\kappa(\mathbf{x_r},\mathbf{x_r})$. The algorithm that arrises from this substitution can perform the transformations with a practically acceptable complexity, whatever the spatial dimension. This transformation (together with a properly chosen kernel function) results in a non-linear feature extraction. The key idea here is that we do not need to know the mapping Φ explicitly; we need only a kernel function $\kappa() : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ for which there exists 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 many good publications about the proper choice of the kernel functions, and also about their theory in general[7]. The two most popular kernels are the following ($p \in \mathbb{N}^+$and $\sigma \in \mathbb{R}^+$):

$$\kappa_1(\mathbf{x},\mathbf{y}) = \left(\mathbf{x}^\top \mathbf{y} + 1\right)^p, \quad \kappa_2(\mathbf{x},\mathbf{y}) = \exp\left(-||\mathbf{x}-\mathbf{y}||^2/\sigma\right). \tag{3}$$

Practically speaking, the original LDA algorithm is executed in a transformed (probably infinite) feature space $\mathcal{H}$ where the kernel function κ gives implicit access to the elements of this space. In the following we present the kernel analogue of LDA by transforming the algorithm $\mathcal{P}$ to $\mathcal{P}'$. Let us consider the following function for a fixed κ, Φ and $\mathcal{H}$.

$$\tau^\Phi(\mathbf{a}) = \frac{\mathbf{a}^\top \mathbf{B}^\Phi \mathbf{a}}{\mathbf{a}^\top \mathbf{W}^\Phi \mathbf{a}}, \qquad \mathbf{a} \in \mathcal{H} \setminus \{\mathbf{0}\}, \tag{4}$$

where the matrices needed for LDA are now given in $\mathcal{H}$:

$$\begin{array}{ll} \mathbf{B}^\Phi = \sum_{j=1}^k \frac{r_j}{r}(\boldsymbol{\mu}_\mathbf{j}^\Phi - \boldsymbol{\mu}^\Phi)(\boldsymbol{\mu}_\mathbf{j}^\Phi - \boldsymbol{\mu}^\Phi)^\top & \mathbf{W}^\Phi = \sum_{j=1}^k \frac{r_j}{r}\mathbf{C}_\mathbf{j}^\Phi \\ \mathbf{C}_\mathbf{j}^\Phi = \frac{1}{r_j}\sum_{f(i)=j}(\Phi(\mathbf{x_i}) - \boldsymbol{\mu}_\mathbf{j}^\Phi)(\Phi(\mathbf{x_i}) - \boldsymbol{\mu}_\mathbf{j}^\Phi)^\top & \boldsymbol{\mu}_\mathbf{j}^\Phi = \frac{1}{r_j}\sum_{f(i)=j}\Phi(\mathbf{x_i}) \end{array} \tag{5}$$

We may suppose without loss of generality that $\mathbf{a} = \sum_{i=1}^r \alpha_i \hat{\Phi}(x_i)$ holds during the search for the stationary points of (4)[4].
Now

$$\begin{aligned} \mathbf{a}^\top \mathbf{B}^\Phi \mathbf{a} = & \left(\sum_{t=1}^r \alpha_t \Phi(\mathbf{x_t})^\top\right) \left[\sum_{j=1}^k \frac{r_j}{r}\left(\left[\frac{1}{r_j}\sum_{f(i)=j}\Phi(\mathbf{x_i})\right] - \left[\frac{1}{r}\sum_{i=1}^r \Phi(\mathbf{x_i})\right]\right)\right. \\ & \left.\left(\left[\frac{1}{r_j}\sum_{f(i)=j}\Phi(\mathbf{x_i})^\top\right] - \left[\frac{1}{r}\sum_{i=1}^r \Phi(\mathbf{x_i})^\top\right]\right)\right] \left(\sum_{s=1}^r \alpha_s \Phi(\mathbf{x_s})\right) \end{aligned} \tag{6}$$

Since $\mathbf{a}^\top \mathbf{B}^\Phi \mathbf{a}$ can also be expressed as $\boldsymbol{\alpha}^\top \mathbf{K}^{B^\Phi} \boldsymbol{\alpha}$ where $\boldsymbol{\alpha} = [\alpha_1,\ldots,\alpha_r]^\top$ and where the matrix $\mathbf{K}^{B^\Phi}$ is of size $r \times r$, and for its elements $\mathbf{K}_{ts}^{B^\Phi}$ with index (t,s) the following holds.:

$$\begin{aligned} \mathbf{K}_{ts}^{B^\Phi} = & \sum_{j=1}^k \frac{r_j}{r}\left(\left[\frac{1}{r_j}\sum_{f(i)=j}\kappa(\mathbf{x_t},\mathbf{x_i})\right] - \left[\frac{1}{r}\sum_{i=1}^r \kappa(\mathbf{x_t},\mathbf{x_i})\right]\right) \\ & \left(\left[\frac{1}{r_j}\sum_{f(i)=j}\kappa(\mathbf{x_i},\mathbf{x_s})\right] - \left[\frac{1}{r}\sum_{i=1}^r \kappa(\mathbf{x_i},\mathbf{x_s})\right]\right) \end{aligned} \tag{7}$$

Then

$$\begin{aligned} \mathbf{a}^\top \mathbf{W}^\Phi \mathbf{a} = & \left(\sum_{t=1}^r \alpha_t \Phi(\mathbf{x_t})^\top\right) \left[\sum_{j=1}^k \frac{1}{r}\sum_{f(i)=j}\left(\Phi(\mathbf{x_i}) - \left[\frac{1}{r_j}\sum_{f(i)=j}\Phi(\mathbf{x_i})\right]\right)\right. \\ & \left.\left(\Phi(\mathbf{x_i})^\top - \left[\frac{1}{r_j}\sum_{f(i)=j}\Phi(\mathbf{x_i})^\top\right]\right)\right] \left(\sum_{s=1}^r \alpha_s \Phi(\mathbf{x_s})^\top\right) \end{aligned} \tag{8}$$

[4] This assumption can be arrived at in several ways, for instance we can decompose an arbitrary vector $\boldsymbol{a}$ into $\mathbf{a_1} + \mathbf{a_2}$, where $\mathbf{a_1}$ is the component of $\mathbf{a}$ which falls in $SPAN(\hat{\Phi}(\mathbf{x_1}),\ldots,\hat{\Phi}(\mathbf{x_r}))$, while $\mathbf{a_2}$ gives the component perpendicular to it. Then from the derivation of $\tau^{\hat{\Phi}}(\mathbf{a})$ it can be proved that $\mathbf{a_2}^\top \mathbf{a_2} = 0$ for stationary points.

We can now express $\mathbf{a}^\top \mathbf{W}^{\boldsymbol{\Phi}} \mathbf{a}$ in the form $\boldsymbol{\alpha}^\top \mathbf{K}^{W^\Phi} \boldsymbol{\alpha}$, where the matrix $\mathbf{K}^{W^\Phi}$ is of size $r \times r$ and

$$\begin{aligned} \mathbf{K}_{ts}^{W^\Phi} = \textstyle\sum_{j=1}^{k} \frac{1}{r} \sum_{f(i)=j} & \left(\kappa(\mathbf{x_t}, \mathbf{x_i}) - \left[\frac{1}{r_j} \textstyle\sum_{f(i)=j} \kappa(\mathbf{x_t}, \mathbf{x_i}) \right] \right) \\ & \left(\kappa(\mathbf{x_i}, \mathbf{x_s}) - \left[\frac{1}{r_j} \textstyle\sum_{f(i)=j} \kappa(\mathbf{x_i}, \mathbf{x_s}) \right] \right) \end{aligned} \tag{9}$$

Combining the above equations we obtain the equality

$$\frac{\mathbf{a}^\top \mathbf{B}^{\boldsymbol{\Phi}} \mathbf{a}}{\mathbf{a}^\top \mathbf{W}^{\boldsymbol{\Phi}} \mathbf{a}} = \frac{\boldsymbol{\alpha}^\top \mathbf{K}^{B^\Phi} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top \mathbf{K}^{W^\Phi} \boldsymbol{\alpha}} \tag{10}$$

This means that (4) can be expressed as dot products of $\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_r})$ and that the stationary points of this equation can be computed using the real eigenvectors[5] of $\left(\mathbf{K}^{W^\Phi}\right)^{-1} \mathbf{K}^{B^\Phi}$. We will use only those eigenvectors which correspond to the m dominant real eigenvalues, denoted by $\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^m$. Consequently, the transformation matrix $\mathbf{A}_\Phi$ of Kernel-LDA is

$$\left[\frac{1}{\theta_1} \sum_{i=1}^{r} \alpha_i^1 \Phi(\mathbf{x_i}), \ldots, \frac{1}{\theta_m} \sum_{i=1}^{r} \alpha_i^m \Phi(\mathbf{x_i}) \right], \quad \theta_k = \left(\sum_{i=1}^{r} \sum_{j=1}^{r} \alpha_i^k \alpha_j^k \kappa(\mathbf{x_i}, \mathbf{x_j}) \right)^{1/2}, \tag{11}$$

where the value of the normalization parameter θ is chosen such that the norm of the column vectors remains unity. For an arbitrary data vector $\mathbf{y}$, new features can be computed via $\mathbf{A_\Phi}^\top \Phi(\mathbf{y}) = \left[\frac{1}{\theta_1} \sum_{i=1}^{r} \alpha_i^1 \kappa(\mathbf{x_i}, \mathbf{y}), \ldots, \frac{1}{\theta_m} \sum_{i=1}^{r} \alpha_i^m \kappa(\mathbf{x_i}, \mathbf{y}) \right]$.

3 Experimental Results

Corpus. For training and testing purposes we recorded samples from 25 speakers, mostly children aged between 8 and 15, but the database used contained some adults too. The speech signals were recorded and stored at a sampling rate of 22050 Hz in 16-bit quality. Each speaker uttered 59 two-syllable Hungarian words of the CVCVC form, where the consonants (C) were mostly unvoiced plosives to ease the detection of the vowels (V). The distribution of the vowels was approximately uniform in the database. Because we decided not to discriminate their long and short versions, we worked with 9 vowels althogether. In the experiments 20 speakers were used for training and 5 for testing.

Feature Sets. The signals were processed in 10 ms frames, the log-energies of 24 critical-bands being extracted using FFT and triangular weighting [5]. The energy of each frame was normalized separately, which means that only the spectral shape was used for classification. Our previous results showed that an additional cosine transform (which would lead to the most commonly used MFCC coefficients) does not affect the

[5] Since in general $\mathbf{K}^{W^\Phi}$ is a positive semidefinite matrix with its determinant sometimes near zero, it can be forced to be invertible using the technique presented in the subsection of LDA. Please see footnote 3 as well.

performance of the classifiers we had intended to apply, so it was omitted. Brief tests showed that neither varying the frame size nor increasing the number of filters gave any significant increase in classifier performance.

In our most basic tests we used only the filter-bank log-energies from the middle frame of the steady-state part of each vowel (“FBLE” set). Then we added the derivatives of these features to model the signal dynamics (“FBLE+Deriv” set). In another experiment we smoothed the feature trajectories to remove the effect of transient noises and disturbances (”FBLE Smooth” set). In yet another set of features we extended the log-energies with the gravity centers of four frequency bands, approximately corresponding to the possible values of the formants. These gravity centers allegedly give a crude approximation of the formants (“FBLA+Grav” set) [1]. Lastly, for the sake of curiosity we performed a test with the feature set of our segmental model(“Segmental” set) [6]. This describes a whole phonemic segment rather than just one frame, it clearly could not be applied in a real-time system. So our aim then was simply to see the advantages of a segmental classifier over a frame-based one.

Classifiers. In all the experiments with *Artificial Neural Nets* (ANN) [2] the well-known three-layer feed-forward MLP networks were employed with the backpropagation learning rule. The number of hidden neurons was equal to the number of features.

In the *Support Vector Machine* (SVM) [7] experiments we always made use of the radial basis kernel function κ_2 (see eq. (3)).

Transformations. In our tests with LDA and Kernel-LDA the eigenvectors belonging to the 16 dominant eigenvalues were chosen as basis vectors for the transformed space and for Kernel-LDA the third-order polynomial kernel κ_1, where $p = 3$ was used (see eq. (3)) .

4 Results and Discussion

Table 1 lists the recognition errors where the rows represent the five feature sets while the columns correspond to the applied transformation and classifier combinations.

On examining the results on the different feature sets we saw that adding the derivative did not increase performance. On the other hand smoothing the trajectories proved beneficial. Most likely a good combination of smoothing and derivation (or even better, RASTA filtering) would give better results.

As regards the gravity center features, they brought about on improvement, but only a slight one. This result accords with our previous experiments [3]. Lastly, the full segmental model clearly performed better than all the frame-based classifiers. This demonstrates the advantage of modeling full phonetic segments over frame-based classification.

When examining the effects of LDA and Kernel-LDA, it can be seen that a non-linear transformation normally performs better in separating the classes than its linear counterpart owing to its larger degree of freedom. One other interesting observation is that although the transformations retained only 16 features, the classifiers attain the same or better scores. Since the computation of LDA is fast, the reduction in the number of features speeds up not only the training but also the recognition phase. As yet, this does not hold for the Kernel-LDA algorithm we currently use, but we are working on a faster implementation.

Finally, as regards the classifiers, SVM consistently outperformed ANN by a few percentage. This can mostly be attributed to the fact that the SVM algorithm cope with overfitting, which is a common problem in ANN training.

	none ANN	none SVM	LDA ANN (16)	LDA SVM (16)	K−LDA ANN (16)	K−LDA SVM (16)
FBLE (24)	26.71 %	22.70 %	25.82 %	24.01 %	24.52 %	21.05 %
FBLE+Deriv (48)	25.82 %	24.01 %	27.30 %	24.34 %	24.34 %	21.21 %
FBLE+Grav (32)	24.01 %	22.03 %	24.67 %	23.85 %	22.87 %	20.72 %
FBLE Smooth (24)	23.68 %	21.05 %	23.03 %	21.87 %	22.70 %	19.90 %
Segmental (77)	19.57 %	19.08 %	20.04 %	18.42 %	18.09 %	17.26 %

Table 1. Recognition errors for the vowel classification task. The numbers in parenthesis correspond to the number of features.

5 Conclusion

Our results show that transforming the training data before learning can definitely increase classifier performance, and also speed up classification. We also saw that a nonlinearized transformation is more effective than the traditional linear version, although they are currently much slower. At present we are working on a sparse data representation scheme that is hoped will give an order of magnitude increase in calculation speed. As regards the classifiers, SVM always performes slightly better than ANN, so we plan to employ it in the future. From the application point of view, our biggest problem at the moment is the feature set. We are looking for more phonetically-based features so as to decrease the classification error, since reliable performance is very important in speech impediment therapy.

References

1. Albesano, D., De Mori, R., Gemello, R., and Mana, F., A study on the effect of adding new dimensions to trajectories in the acoustic space, *Proc. of EuroSpeech'99,* pp. 1503-1506, 1999.
2. Bishop, C. M., *Neural Networks for Pattern Recognition,* Oxford University Press, 1995.
3. Kocsor, A., Tóth, L., Kuba, A. Jr., Kovács, K., Jelasity, M., Gyimóthy, T., and Csirik, J., A Comparative Study of Several Feature Transformation and Learning Methods for Phoneme Classification, *Int. Journal of Speech Technology,* Vol. 3., No. 3/4, pp. 263-276, 2000.
4. Mika, S., Rätsch, G., Weston, J., Schölkopf, B., and Müller, K.-R., Fisher Discriminant Analysis with Kernels, In Hu, Y.-H., Larsen, E., Wilson, E. and Douglas, S., editors, *Neural Networks for Signal Processing IX,* pages 41-48, IEEE, 1999.
5. Rabiner, L. R., Juang, B.-H., *Fundamentals of Speech Recognition,* Englewood Cliffs, NJ, Prentice Hall, 1993.
6. Toth, L., Kocsor, A., and Kovács, K., A Discriminative Segmental Speech Model and Its Application to Hungarian Number Recognition, In *Sojka, P. et al.(eds.), Text, Speech and Dialogue, Proc. of TSD'2000,* Springer Verlag LNAI Series, vol. 1902, pp. 307-313, 2000.
7. Vapnik, V. N., *Statistical Learning Theory,* John Wiley & Sons Inc., 1998.