Application of Kernel-Based Feature Space Transformations and Learning Methods to Phoneme Classification [*]

András Kocsor and László Tóth
Research Group on Artificial Intelligence
of the Hungarian Academy of Sciences and University of Szeged
H-6720 Szeged, Aradi vértanúk tere 1., Hungary
*(*kocsor@inf.u-szeged.hu, tothl@inf.u-szeged.hu*)*

Abstract. This paper examines the applicability of some learning techniques to the classification of phonemes. The methods tested were artificial neural nets (ANN), support vector machines (SVM) and Gaussian mixture modeling (GMM). We compare these methods with a traditional hidden Markov phoneme model (HMM), working with the linear prediction-based cepstral coefficient features (LPCC). We also tried to combine the learners with linear/nonlinear and unsupervised/supervised feature space transformation methods such as principal component analysis (PCA), independent component analysis (ICA), linear discriminant analysis (LDA), springy discriminant analysis (SDA) and their nonlinear kernel-based counterparts. We found that the discriminative learners can attain the efficiency of HMM, and that after the transformations they can retain the same performance in spite of the severe dimension reduction. The kernel-based transformations brought only marginal improvements compared to their linear counterparts.

Keywords: phoneme classification, feature extraction, principal component analysis, independent component analysis, discriminant analysis, kernel-based methods

1. Introduction

Automatic speech recognition is a special pattern classification problem which aims to mimic the perception and processing of speech in humans. For this reason it clearly belongs to the fields of machine learning (ML) and artificial intelligence (AI). For historical reasons, however, it is mostly ranked as a sub-field of electrical engineering, with its own unique technologies, conferences and journals. In the last two decades the dominant method for speech recognition has been the hidden Markov modeling (HMM) approach. Meanwhile, the theory of machine learning has developed considerably and now has a wide variety of learning and classification algorithms for pattern recognition problems. The goal of this paper is to study the applicability of some of these methods to phoneme classification, making use of so-called feature space transformation methods applied prior to learning to improve classification rates. In essence, this article deals with the neural network (ANN), support vector machine (SVM) and Gaussian Mixture modeling (GMM) learning methods and with feature space transformations principal component analysis (PCA), independent component analysis (ICA), linear discriminant analysis (LDA), springy discriminant analysis (SDA) and their nonlinear kernel-based counterparts [2, 3, 19–22, 26]. We compare the performance of the learners with that of HMM on the same feature set, namely the so-called linear prediction-based cepstral coefficients (LPCC).

The structure of the paper is as follows. First, we provide a short review of the phoneme classification problem itself and suggest some possible solutions. Then we briefly describe the acoustic features that were applied in the experiments and review the linear and nonlinear feature space transformation methods used. The final part of the paper discusses aspects of the experiments, especially the advantages and drawbacks of the learning methods compared, the effectiveness of each feature space transformation method and, of course, the results obtained.

2. The Task of Phoneme Classification

Speech recognition is a pattern classification problem in which a continuously varying signal has to be mapped to a string of symbols (the phonetic

[*] This work was supported under the contract IKTA No. 2001/055 from the Hungarian Ministry of Education.

transcription). Speech signals display so many variations that attempts to build *knowledge-based speech recognizers* have mostly been abandoned. Currently researchers tackle speech recognition only with statistical pattern recognition techniques. Here, however a couple of special problems arise that have to be dealt with. The first one is the question of the recognition unit. The basis of the statistical approach is the assumption that we have a finite set of units (in other words, classes), the distribution of which is modelled statistically from a large set of training examples. During recognition an unknown input is classified as one of these units, using some kind of similarity measure. Since the number of possible sentences or even words is potentially infinite, some sort of smaller recognition units have to be chosen in a general speech recognition task. The most commonly used unit of this kind is the phoneme, thus this paper deals with the classification problem of phonemes.

The other special problem is that the length of the units may vary, that is utterances get warped along the time axis. The only known way of solving this is to perform a search in order to locate the most probable mapping between the signal and the possible transcriptions. Normally a depth-first search is applied (implemented with dynamic programming), but a breadth-first search with a good heuristic is also viable.

3. Hidden Markov and Segmental Phoneme Modeling

Hidden Markov Models [23] synchronously handle both the problems mentioned above. Each phoneme in the speech signal is given as a series of observation vectors $O = \mathbf{o}_1, \ldots, \mathbf{o}_T$, and one has one model for each unit of recognition c. These models eventually return a class-conditional likelihood $P(O|c)$. The models are composed of states, and for each state we model the probability that a given observation vector belongs to ("was omitted by") this state. Time warping is handled by state transition probabilities, that is the probability that a certain state follows the given state. The final "global" probability is obtained as the product of the proper omission and state-transition probabilities.

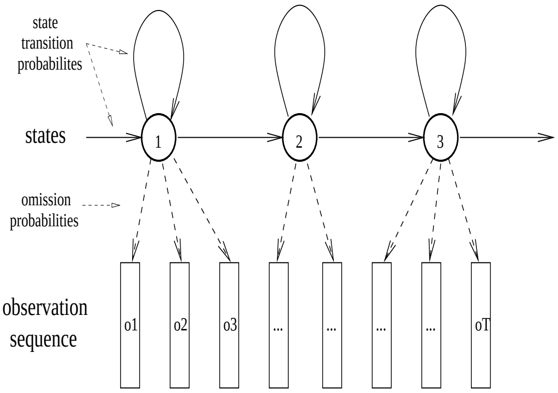

Figure 1. The three-state left-to-right phoneme HMM.

When applied to phoneme recognition, the most common state topology is the three-state left-to-right model (see Fig.1). We use three states because the first and last parts of a phoneme are usually different from the middle due to coarticulation. This means that in a sense we do not really model phonemes but rather phoneme thirds.

Because the observation vectors usually have continuous values the state omission probabilities have to be modeled as multidimensional likelihoods. The usual procedure is to employ a mixture of weighted Gaussian distributions of the form

$$p(\mathbf{o_j}) = \sum_{i=1}^{k} \alpha_i \mathcal{N}(\mathbf{o_j}, \boldsymbol{\mu}_i, C_i), \qquad (1)$$

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

A possible alternative to HMM are the Stochastic Segmental Models. The more sophisticated segmental techniques fit parametric curves to the feature trajectories of the phonemes [24]. There is, however, a much simpler methodology [13] that applies non-uniform smoothing and sampling in order to parametrize any phoneme with the same number of features, independent of its length. The advantage of this uniform parametrization is that it allows us to apply any sort of machine learning algorithm to the phoneme classification task. This is why we chose this type of segmental modeling for the experiments of this paper and also for our speech recognition system [30].

4. Generative and Discriminative Modeling

Hidden Markov Models describe the class conditional likelihoods $P(O|c)$. These type of models are called *generative* because they model the probability that an observation O was generated by a class c. However, the final goal of classification is to find the most probable class c. We can compute the posterior probabilities $P(c|O)$ from $P(O|c)$ using Bayes' law. Another approach is to model the posteriors directly. This is how *discriminative* learners work. Instead of describing the distribution of the classes, these methods model the surfaces that separate the classes and usually perform slightly better than generative models. In this paper we applied one generative and two discriminative learners to classify the phonemes based on the segmental features. The generative one was the *Gaussian Mixture Model* [9]. This is virtually the same as the state omission formula of HMM (1), but in this case it is used to parametrize a whole phoneme and not frame-based observation vectors. From the family of discriminative learners we chose to experiment with the now traditional *Artificial Neural Networks* [4], and a relatively new technology called *Support Vector Machines.* Rather than describing this method in detail here we refer the interested reader to the overview in [32].

5. Evaluation Domain

The feature space transformations and the classification techniques were compared using a relatively small 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 speakers 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 labelled. Since the corpus contained only numbers we had samples of only 32 phones, which is approximately two thirds of the Hungarian phoneme set. As some of these labels represented only allophonic variations of the same phoneme, some labels were fused together, hence in practice we 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. The training and test sets contained 5616 and 1692 data lines, respectively.

6. Frame-Based and Segmental Features

There are numerous methods for obtaining representative initial feature vectors from speech data [23], but their common property is that they are all extracted from 20-30 ms chunks or ”frames” of the signal in 5-10 ms time steps. The HMM system employed in our experiments was the FlexiVoice speech engine [29] that works with the lpc-based cepstral coefficients [23]. Because of this we conducted all the experiments with this feature set. To be more precise, 17 LPCC coefficients (including the zeroth one) were extracted from 30 ms frames. In the case of the HMM system the derivatives of these were used as well, so *a speech frame* was characterized by 34 features altogether.

All the other classifiers were tested within the framework of the OASIS speech recognizer [30]. This is a segment-based recognizer that presumes that any phonemic segment is described with the same number of parameters. These segmental features were obtained as follows. We averaged the 17 LPCC coefficients over segment thirds, which resulted in $3 * 17 = 51$ features per phoneme. As mentioned earlier, besides the frames themselves HMM also uses frame-based derivatives to model the spectral dynamics. To incorporate something similar in our model, we simply took the differences of the neighbouring segmental averages. We applied the same computation too at the phoneme boundaries, and thus we obtained $4*17 = 68$ derivative like features. Finally, the phoneme length was included as a very important feature. This means that altogether 120 features were used to describe *a complete phoneme.*

When we compare this simple segmental model to HMM it is quite clear that the latter is much more involved. HMM makes use of all the signal frames directly, while the segmental model drastically smooths this information by representing them only by three averages. Also, HMM has the flexibility to model the possible variance of the position of the phonemic start and end phases, while in the segmental model the averages are always taken over rigid one-third divisions. One goal of this paper is to show that even with this primitive segmental model the classification performance of HMM can be reached. In other experiments we found that HMM can be outperformed by the segmental model with the addition of some more complex segmental features [18]. In this paper, however, we refrained from using these because we just wanted to keep the segmental representation as similar to the HMM one as possible.

In all the experiments we worked with the LPCC coefficients because we wanted to have the segmental and HMM inputs as similar as possible. In earlier research, however, we found that this representation is definitely not optimal for our system. Just to show the capabilities of the segmental model, we also report findings obtained via the bark-scaled filterbank log-energies (FBLE). This means that the signal is decomposed with a special filterbank and the energies in these filters are used to parameterize speech on a frame-by-frame basis. The filters were approximated via Fourier analysis with a triangular weighting, as described in [23]. The segmental features were calculated from FBLE in the same way as from LPCC.

7. Linear Feature Space Transformation

Before executing a learning algorithm, additional vector space transformations may be applied on the initial features. The reason for doing this is twofold. Firstly they can improve classification performance and, secondly, they can also reduce the dimensionality of the data.

Without loss of generality we shall assume that as a realization of multivariate random variables, there are n-dimensional real attribute vectors in a compact set $\mathcal{X}$ over $\mathbb{R}^n$ describing objects in a certain domain, and that we have a finite $n \times k$ sample matrix $X = [\mathbf{x}_1, \ldots, \mathbf{x}_k]$ containing k random observations. Actually, $\mathcal{X}$ constitutes our initial feature space and X is the input data for the algorithm, which determines the linear transformation itself. The $m \times n$ $(n \leq m)$ matrix of the linear transformation - which may inherently include a dimension reduction - will be denoted by V. The result $V\mathbf{z}$ of the transformation applied to an arbitrary vector $\mathbf{z} \in \mathcal{X}$ will be denoted by $\mathbf{z}^*$.

With the linear feature space transformation methods we search for an optimal (in some cases orthogonal) matrix V, where the precise definition of optimality can vary from method to method. Although it is possible to define functions that measure the optimality of all the m directions (i.e. the raw vectors of V) *together*, we find each particular direction of the optimal transformations *one-by-one*, employing a $\tau : \mathbb{R}^n \rightarrow \mathbb{R}$ objective function for each direction separately. Intuitively, if larger values of τ indicate better directions and the chosen m directions need to be independent in some ways, then choosing stationary points that have the m largest function values is a reasonable strategy. Obtaining the above stationary points of a general objective function is a difficult global optimization problem. But if τ is defined by Rayleigh quotient formulae the solution is easy and fast when formulated as a simple eigenvalue problem. Actually, this approach offers a unified view of the linear transformation methods discussed in this paper.

Finally, let us assume as well that we have r classes and an indicator function

$$\mathcal{L} : \{1, \ldots, k\} \rightarrow \{1, \ldots, r\}, \tag{2}$$

where $\mathcal{L}(i)$ gives the class label of the sample $\mathbf{x}_i$. Let k_j further denote the number of vectors associated with label j in the sample data.

The two types of feature vector space transformations (supervised or unsupervised) can be distinguished by whether they utilize an indicator function or not.

7.1. Principal Component Analysis

Principal component analysis [16] is a ubiquitous unsupervised technique for data analysis and dimension reduction. Normally in PCA the objective function τ for selecting new directions is defined by

$$\tau(\mathbf{v}) = \frac{\mathbf{v}^\top C\mathbf{v}}{\mathbf{v}^\top\mathbf{v}}, \tag{3}$$

where $C = E\{(\mathbf{x} - E\{\mathbf{x}\})(\mathbf{x} - E\{\mathbf{x}\})^\top\}$ is the sample covariance matrix, where E denotes the mean value. It is easy to see that (3) defines $\tau(\mathbf{v})$ as the variance of the centralized sample vectors $\mathbf{x}_1 - E\{\mathbf{x}\}, \ldots, \mathbf{x}_k - E\{\mathbf{x}\}$ projected onto vector $\mathbf{v}$. So this method prefers directions having a large variance. It can be shown as well that stationary points of (3) correspond to the right eigenvectors of the sample covariance matrix C where the eigenvalues form the corresponding optimum values. If we assume that the eigenpairs of C are $(\mathbf{c}_1, \lambda_1), \ldots, (\mathbf{c}_k, \lambda_k)$ and $\lambda_1 \geq \cdots \geq \lambda_k$, then the transformation matrix V will be $[\mathbf{c}_1, \ldots, \mathbf{c}_m]^\top$, i.e. the eigenvectors with the largest m eigenvalues. Notice that since the sample covariance matrix is symmetric and positive semidefinite its eigenvalues are nonnegative and its eigenvectors are orthogonal. Moreover, after applying the above orthogonal transformation V on the sample data X, we find that VX is uncorrelated, i.e. its covariance matrix is diagonal with $\lambda_1, \ldots, \lambda_m$ being in the diagonal.

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the PCA transformation can be done using $\mathbf{z}^* = V\mathbf{z}$. But if the output data needs to be centralized we can apply $V(\mathbf{z} - E\{\mathbf{x}\})$ as well.

7.2. Independent Component Analysis

Independent Component Analysis [7, 10, 14, 15] is a general purpose statistical method that originally arose from the study of blind source separation (BSS). A typical BSS problem is the cocktail-party problem where several people are speaking simultaneously in the same room and several microphones record a mixture of speech signals. The task is to separate the voices of different speakers using the recorded samples. Another application of ICA is unsupervised feature extraction, where the aim is to linearly transform the input data into uncorrelated components, along which the distribution of the sample set is the least Gaussian. The reason for this is that along these directions the data is supposedly easier to classify. For optimal selection of the independent directions several objective function were defined using approximately equivalent approaches. Here we follow the way proposed by Hyvärinen [10, 14, 15]. Generally speaking, we expect these functions to be non-negative and have a zero value for the Gaussian distribution. Negentropy is a useful measure having just this property, which is used for assessing non-Gaussianity (i.e. the least Gaussianity). Since obtaining this quantity via its definition is computationally rather difficult, a simple easily-computable approximation is normally employed. The negentropy of a variable η with zero mean and unit variance is estimated by using the formula

$$J_G(\eta) \approx (E\{G(\eta)\} - E\{G(\nu)\})^2 \tag{4}$$

where $G : \mathbb{R} \to \mathbb{R}$ is an appropriate non-quadratic function, E again denotes the expected value and ν is a standardized Gaussian variable. The following three choices of G are conventionally used:

$$\begin{aligned} G_1(\eta) &= \eta^4, \\ G_2(\eta) &= \log(\cosh(\eta)), \\ G_3(\eta) &= -\exp(-\eta^2/2). \end{aligned} \tag{5}$$

It should be mentioned that in (4) the expected value of $G(\nu)$ is a constant, its value only depending on the selected function (e.g. $E\{G_1(\nu)\} = 3$). In Hyvärinen's FastICA algorithm for the selection of a new direction $\mathbf{v}$ the following τ objective function is used:

$$\tau_G(\mathbf{v}) = (E\{G(\mathbf{v} \cdot \mathbf{x})\} - E\{G(\nu)\})^2, \tag{6}$$

which can be obtained by replacing η in the negentropy approximant (4) with $\mathbf{v} \cdot \mathbf{x}$, the dot product of the direction $\mathbf{v}$ and sample $\mathbf{x}$. FastICA is an approximate Newton iteration procedure for the local optimization of the function $\tau_G(\mathbf{v})$. Before running

FastICA, however, the raw input data X must first be preprocessed – by centering and whitening it[1].

Centering. An essential step is to shift the original sample set $\mathbf{x}_1, \ldots, \mathbf{x}_k$ with its mean $E\{\mathbf{x}\}$, to obtain data $\mathbf{x}'_1 = \mathbf{x}_1 - E\{\mathbf{x}\}, \ldots, \mathbf{x}'_k = \mathbf{x}_k - E\{\mathbf{x}\}$, with a mean of zero.

Whitening. The goal of this step is to transform the centered samples $\mathbf{x}'_1, \ldots, \mathbf{x}'_k$ via an orthogonal transformation Q into vectors $\hat{\mathbf{x}}_1 = Q\mathbf{x}'_1, \ldots, \hat{\mathbf{x}}_k = Q\mathbf{x}'_k$ where the covariance matrix $\hat{C} = E\{\hat{\mathbf{x}}\hat{\mathbf{x}}^\top\}$ is the unit matrix. Since standard principal component analysis [16] transforms the covariance matrix into a diagonal form, where the diagonal elements are the eigenvalues of the original covariance matrix $E\{\mathbf{x}'\mathbf{x}'^\top\}$, it only remains to transform each diagonal element to one. Now if we assume that the eigenpairs of $E\{\mathbf{x}'\mathbf{x}'^\top\}$ are $(\mathbf{c}_1, \lambda_1), \ldots, (\mathbf{c}_k, \lambda_k)$ and $\lambda_1 \geq \ldots \geq \lambda_k$, the transformation matrix Q will take the form $[\mathbf{c}_1\lambda_1^{-1/2}, \ldots, \mathbf{c}_m\lambda_m^{-1/2}]^\top$. If m is less than k a dimensionality reduction is employed.

Properties of the preprocessing stage. Firstly, after centering and whitening for every normalized $\mathbf{v}$ the mean of $\mathbf{v} \cdot \hat{\mathbf{x}}_1, \ldots, \mathbf{v} \cdot \hat{\mathbf{x}}_k$ is set to zero, and its variance is set to one. Actually we need this since (4) requires that η should have a zero mean and variance of one hence, with the substitution $\eta = \mathbf{v} \cdot \hat{\mathbf{x}}$, the projected data $\mathbf{v} \cdot \hat{\mathbf{x}}$ must also have this property. Secondly, for any matrix W the covariance matrix $\hat{C}_W$ of the transformed preprocessed points $W\hat{\mathbf{x}}_1, \ldots, W\hat{\mathbf{x}}_k$ will remain a unit matrix if and only if W is orthogonal, since

$$\begin{aligned} \hat{C}_W &= E\{W\hat{\mathbf{x}}(W\hat{\mathbf{x}})^\top\} = WE\{\hat{\mathbf{x}}\hat{\mathbf{x}}^\top\}W^\top \\ &= WIW^\top = WW^\top. \end{aligned} \tag{7}$$

FastICA. After preprocessing, this method looks for a new orthogonal base W for the preprocessed data, where the values of the non-Gaussianity measure τ_G for the base vectors are large[2]. The following pseudo-code is helpful in understanding how the technique works[3]:

```
% The input for this algorithm is the
% sample matrix $X$ and the nonlinear
% function $G$, while the output is the
% transformation matrix $W$. The first
% and second order derivatives of $G$
% are denoted by $G'$ and $G''$.
procedure FastICA($X, G$);
  % centering & whitening
  $\mathbf{x}' = \mathbf{x} - E\{\mathbf{x}\}$;  $\hat{\mathbf{x}} = Q\mathbf{x}'$;
  % initialization
  let $W_0$ be a random $m \times m$ matrix;
  $W_0 = (W_0 W_0^\top)^{-1/2} W_0$;
  $i = 0$;
  % approximate Newton iteration
  While $W$ has not converged;
    for $j = 1$ to $m$
      let $\mathbf{s}_j$ be the $jth$ raw vector of $W_i$;
      $\mathbf{w}_j = E\{\hat{\mathbf{x}} G'(\mathbf{s}_j \cdot \hat{\mathbf{x}})\} - E\{G''(\mathbf{s}_j \cdot \hat{\mathbf{x}})\}\mathbf{s}_j$;
    end;
    $i = i + 1$;
    $W_i = [\mathbf{w}_1, \ldots, \mathbf{w}_m]^\top$;
    $W_i = (W_i W_i^\top)^{-1/2} W_i$;
  do
End procedure
```

In the pseudo-code $(W_i W_i^\top)^{-1/2} W_i$ means a symmetric decorrelation, where $(W_i W_i^\top)^{-1/2}$ can be readily obtained from its eigenvalue decomposition. If $W_i W_i^\top = EDE^\top$, then $(W_i W_i^\top)^{-1/2}$ is equal to $ED^{-1/2}E^\top$.

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the ICA transformation can be done using $\mathbf{z}^* = WQ\mathbf{z}$. Here W denotes the orthogonal transformation matrix we obtained as the output from FastICA, while Q is the matrix obtained from whitening. Much like that in PCA, if we require that the output data be centralized then $\mathbf{z}^* = WQ(\mathbf{z} - E\{\mathbf{x}\})$.

[1] There are many other iterative methods for performing Independent Component Analysis, some of these similar to FastICA do require centering and whitening, while others do not. In general, experience has taught us that all these algorithms should converge faster on centered and whitened data, even those which do not really require it.

[2] Note that since the data remains whitened after an orthogonal transformation, ICA can be considered an extension of PCA.

[3] `MatLab` code available in [10].

7.3. Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a traditional supervised feature extraction method [11] which has proved to be one of the most successful preprocessing techniques for classification[4]. The goal of LDA is to find a new (not necessarily orthogonal) basis for the data which provides the optimal separation between groups of sample points (classes). In order to define the transformation matrix of LDA we first define the objective function $\tau : \mathbb{R}^n \to \mathbb{R}$ which depends not only on the sample data X, but also on the indicator function $\mathcal{L}$ owing to the supervised nature of this method. Let us define

$$\tau(\mathbf{v}) = \frac{\mathbf{v}^\top B\mathbf{v}}{\mathbf{v}^\top W\mathbf{v}}, \qquad \mathbf{v} \in \mathbb{R}^n \setminus \{\mathbf{0}\}, \tag{8}$$

where B is the *Between-class Scatter Matrix*, while W is the *Within-class Scatter Matrix*. Here the *Between-class Scatter Matrix B* shows the scatter of the class mean vectors $\mathbf{m}_j$ around the overall mean vector $\mathbf{m}$:

$$\begin{array}{rcl} B &=& \sum_{j=1}^{r} \frac{k_j}{k}(\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^\top \\ \mathbf{m} &=& \frac{1}{k}\sum_{i=1}^{k} \mathbf{x}_i \\ \mathbf{m}_j &=& \frac{1}{k_j}\sum_{\mathcal{L}(i)=j} \mathbf{x}_i \end{array} \tag{9}$$

The *Within-class Scatter Matrix* W represents the weighted average scatter of the covariance matrices C_j of the sample vectors having label j:

$$\begin{array}{rcl} W &=& \sum_{j=1}^{r} \frac{k_j}{k} C_j \\ C_j &=& \frac{1}{k_j}\sum_{\mathcal{L}(i)=j} (\mathbf{x}_i - \mathbf{m}_j)(\mathbf{x}_i - \mathbf{m}_j)^\top. \end{array} \tag{10}$$

$\tau(\mathbf{v})$ is large when its nominator is large and its denominator is small or, equivalently, when the within-class averages of the sample projected onto $\mathbf{v}$ are far from each other and the variance of the classes is small. The larger the value of $\tau(\mathbf{v})$ the farther the classes will be spaced and the smaller their spreads will be. It can be easily shown that stationary points of (8) correspond to the right eigenvectors of $W^{-1}B$, where the eigenvalues form the corresponding function values. Since $W^{-1}B$ is not necessarily symmetrical and its rank is at most the class number minus one $(r-1)$, the number of its real, not necessarily orthogonal eigenvectors can be less than r. Besides this, numerical problems can arise during the computation of W^{-1} if $det(W)$ is near zero. The most probable cause for this could be the redundancy of feature components. But we know W is positive semidefinite. So if we add a small positive constant ϵ to its diagonal, that is we work with $W + \epsilon I$ instead of 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 (8). If we assume that the real eigenvectors with the largest $m(< r)$ real eigenvalues of $W^{-1}B$ are $\mathbf{c}_1, \ldots, \mathbf{c}_m$, then the transformation matrix V will be $[\mathbf{c}_1, \ldots, \mathbf{c}_m]^\top$.

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the LDA transformation can be done using $\mathbf{z}^* = V\mathbf{z}$.

7.4. Springy Discriminant Analysis

We saw that LDA can become numerically unstable because of the invertibility problem of the *Within-class Scatter Matrix.* Furthermore, the non-orthogonality of the resulting transformation matrix may prove disadvantageous. These issues give rise to the need for an objective function τ that leads to an unsupervised transformation which yields similar results to LDA, but is orthogonal and avoids the numerical problems mentioned [21]. To do this, let us first define the matrix Θ as

$$[\Theta]_{ij} = \begin{cases} -1, & \text{if } \mathcal{L}(i) = \mathcal{L}(j) \\ 1, & \text{otherwise} \end{cases} \qquad i, j = 1, \ldots, k. \tag{11}$$

Then, let the function δ be defined by

$$\delta(\mathbf{v}) = \sum_{i,j=1}^{k} \left((\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{v} \right)^2 [\Theta]_{ij} \tag{12}$$

It is easy to see that the value of δ is largest when those components of the elements of the same class that fall in the given direction $\mathbf{v} \in \mathbb{R}^n$, $\|\mathbf{v}\| = 1$ are close, and the components of different classes are far at the same time. This can be viewed as if the data point (vectors) of the same class were

[4] One should note here that the technique can be directly used for classification as well.

interconnected by springs, and the points of different classes by "anti-springs". The goal of the transformation is to minimize the system's potential as a function of direction $\mathbf{v}$. The name of the transformation stems from this physical analogy. Now by the introduction of the matrix

$$D = \sum_{i,j=1}^{k} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top [\Theta]_{ij} \qquad (13)$$

we immediately obtain that $\delta(\mathbf{v}) = \mathbf{v}^\top D\mathbf{v}$. Based on this, the objective function τ can be defined as the Rayleigh quotient

$$\tau(\mathbf{v}) = \frac{\delta(\mathbf{v})}{\mathbf{v}^\top\mathbf{v}} = \frac{\mathbf{v}^\top\mathbf{D}\mathbf{v}}{\mathbf{v}^\top\mathbf{v}}. \qquad (14)$$

Obviously, the optimization of τ leads to the eigenvalue decomposition of D, just like in the case of PCA. Because D is symmetric, its eigenvalues are real and its eigenvectors are orthogonal. For this reason, the matrix V of the transformation is defined using those eigenvectors corresponding to the m dominant eigenvalues of D.

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the SDA transformation can be made using $\mathbf{z}^* = V\mathbf{z}$.

8. Kernel-based Feature Vector Transformations

The approach of feature extraction could be either linear or nonlinear, but it seems there is a technique (which is most topical nowadays) that is, in some sense, breaking down the barrier between the two types. The key idea behind the kernel technique was originally presented in [1] and was again applied in connection with the general purpose Support Vector Machine [5, 6, 28, 31, 32], which was followed by other kernel-based methods [2, 3, 19–22, 26, 27, 25]. To put the kernel idea in a nutshell, we might explain it as follows. If some method uses only the pairwise scalar product $(\mathbf{x_i}\cdot\mathbf{x_j})$ of its input vectors during its computations then, just by altering the scalar product operation in a proper way, we can create a non-linear version of the method. The effect of the replacement of the operation is that the original method will implicitly be performed in a space of more (possibly even infinite) dimensions, and thus with a higher degree of freedom. In the following this notion is also used to derive nonlinear counterparts of PCA, ICA, LDA and SDA.

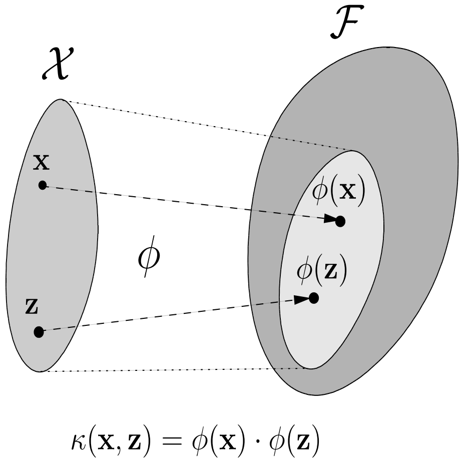

Figure 2. The "kernel-idea". The dot product in the kernel feature space $\mathcal{F}$ is defined implicitly.

We recall that we have a sample matrix $X = [\mathbf{x_1}, \ldots, \mathbf{x_k}]$ containing k random observations and that we have an indicator function $\mathcal{L}$ which returns the class label of each sample vector.

8.1. Kernel-Induced Feature Spaces

A 'kernel' is a continuous function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (see Fig. 2) for which there exists an $\mathcal{F}$ inner product space as a representation space and a map $\phi : \mathcal{X} \to \mathcal{F}$ such that for all

$$\mathbf{x}, \mathbf{z} \in \mathcal{X} \quad \kappa(\mathbf{x},\mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \qquad (15)$$

This definition allows us to perform calculations in the $\mathcal{F}$ space in an implicit way, by substituting the scalar product operation with its corresponding kernel version. Usually $\mathcal{F}$ is called the kernel feature space and ϕ is the feature map. Knowing ϕ explicitly – and, consequently, knowing $\mathcal{F}$ – is not necessary. We need only define the kernel function, which then ensures an implicit evaluation. The construction of an appropriate kernel function (i.e. when such a function ϕ exists) is a non-trivial problem, but there are many good suggestions about the sorts of kernel functions which might be adopted along with some background theory [8, 12, 17]. From the functions available, the two most popular are the *polynomial kernel* κ_1 and the *Gaussian RBF kernel* κ_2:

$$\begin{array}{rclr} \kappa_1(\mathbf{x},\mathbf{z}) & = & (\mathbf{x}\cdot\mathbf{z}+1)^d\,, & d\in\mathbb{N}, \\ \kappa_2(\mathbf{x},\mathbf{z}) & = & \exp\left(-\|\mathbf{x}-\mathbf{z}\|^2/\gamma\right), & \gamma\in\mathbb{R}_+. \end{array} \tag{16}$$

8.2. PCA in Kernel Feature Spaces

Having chosen a proper κ kernel function for which

$$\kappa(\mathbf{x},\mathbf{z}) = \phi(\mathbf{x})\cdot\phi(\mathbf{z}), \qquad \mathbf{x},\mathbf{z}\in\mathbb{R}^n,$$

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

First, let us examine the form of the τ objective function in the kernel feature space $\mathcal{F}$:

$$\tau(\mathbf{v}) = \frac{\mathbf{v}^\top\mathcal{C}\mathbf{v}}{\mathbf{v}^\top\mathbf{v}}, \qquad \mathbf{v}\in\mathcal{F}\setminus\{\mathbf{0}\}, \tag{17}$$

where $\mathcal{C}$ is the covariance matrix of the sample $\phi(\mathbf{x}_1),\ldots,\phi(\mathbf{x}_k)$:

$$\mathcal{C} = E\{(\phi(\mathbf{x})-E\{\phi(\mathbf{x})\})\,(\phi(\mathbf{x})-E\{\phi(\mathbf{x})\})^\top\}. \tag{18}$$

Much like the PCA approach, we define the Kernel-PCA transformation based on the stationary points of (17), which are given as the eigenvectors of the symmetric positive semidefinite matrix $\mathcal{C}$. However, since this matrix is of the form

$$\mathcal{C} = \sum_{i,j}^{k} c_{ij}\phi(\mathbf{x}_i)\phi(\mathbf{x}_j)^\top, \tag{19}$$

we can suppose the following equation holds during the analysis of the stationary points:

$$\mathbf{v} = \sum_{i=1}^{k}\alpha_i\phi(x_i). \tag{20}$$

We can arrive at this assumption in many ways, e.g. we can decompose an arbitrary vector $\mathbf{v}$ into vectors $\mathbf{v}_1+\mathbf{v}_2$, where $\mathbf{v_1}$ gives that component of $\mathbf{v}$ which falls in $SPAN(\phi(\mathbf{x}_1),\ldots,\phi(\mathbf{x}_k))$, while $\mathbf{v}_2$ gives the component perpendicular to it. Then from the derivation of (17) we see that $\mathbf{v}_2^\top\mathbf{v}_2 = 0$ for the stationary points. Based on the above assumption the variational parameters of τ can be the vector $\boldsymbol{\alpha}$ instead of $\mathbf{v}$:

$$\tau(\boldsymbol{\alpha}) = \frac{\left(\sum_{i=1}^{k}\alpha_i\phi(\mathbf{x}_i)^\top\right)\mathcal{C}\left(\sum_{j=1}^{k}\alpha_j\phi(\mathbf{x}_j)\right)}{\left(\sum_{i=1}^{k}\alpha_i\phi(\mathbf{x}_i)^\top\right)\left(\sum_{j=1}^{k}\alpha_j\phi(\mathbf{x}_j)\right)}. \tag{21}$$

It is easy to see that

$$\tau(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^\top\frac{1}{k}K(I-\bar{1})K\boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}}, \tag{22}$$

where $[K]_{ij} = \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j) = \kappa(\mathbf{x}_i,\mathbf{x}_j)$ is a Gram matrix and $[\bar{1}]_{ij} = 1/k$.

After differentiating (22) with respect to $\boldsymbol{\alpha}$ we see that the stationary points are the solution vectors of the general eigenvalue problem

$$\frac{1}{k}K(I-\bar{1})K\boldsymbol{\alpha} = \lambda K\boldsymbol{\alpha}, \tag{23}$$

which is equivalent to the problem

$$\frac{1}{k}(I-\bar{1})K\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}. \tag{24}$$

Although the matrix $(I-\bar{1})K$ is not symmetric, its eigenvalues are real and non-negative, and those eigenvectors that correspond to positive eigenvalues are orthogonal. In fact the best approach is to solve the following symmetric eigenproblem, where the positive eigenvalues and the corresponding eigenvectors are equal to those obtained from (24)

$$\frac{1}{k}(I-\bar{1})K(I-\bar{1})\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}. \tag{25}$$

We note that we would have arrived at the same eigenproblem had we presumed that

$$\mathbf{v} = \sum_{i=1}^{k}\alpha_i\left(\phi(\mathbf{x}_i)-E\{\phi(\mathbf{x})\}\right) \tag{26}$$

instead of (20). With this assumption we would have obtained the function

$$\tau(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^\top\frac{1}{k}(I-\bar{1})K(I-\bar{1})(I-\bar{1})K(I-\bar{1})\boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top(I-\bar{1})K(I-\bar{1})\boldsymbol{\alpha}}. \tag{27}$$

Now let the m positive dominant eigenvalues of $\frac{1}{k}(I-\bar{1})K(I-\bar{1})$ be denoted by $\lambda_1 \geq \ldots \geq \lambda_m$ and the corresponding eigenvectors be $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m$. Then the orthogonal matrix of the transformation we need can be calculated like so:

$$\begin{aligned} V &= A\bar{X}, \\ A &= [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m]^\top, \\ \bar{X} &= [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_k)]^\top. \end{aligned} \tag{28}$$

Of course the norm of the eigenvectors alpha can easily chosen such that the norm of the column vectors of the transformation matrix $V = A\bar{X}$ becomes 1.

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the Kernel-PCA transformation can be done using $\mathbf{z}^* = V\phi(\mathbf{z}) = A\bar{X}\phi(\mathbf{z}) = A\kappa(X, \mathbf{z})$, where $\kappa(X, \mathbf{z})$ is a shorthand notation for

$$[\kappa(\mathbf{x}_1, \mathbf{z}), \ldots, \kappa(\mathbf{x}_k, \mathbf{z})]^\top. \tag{29}$$

8.3. ICA in Kernel Feature Spaces

In this section we derive the kernel counterpart of *FastICA* [20]. To this end, let the inner product be implicitly defined by the kernel function κ in $\mathcal{F}$ with associated transformation ϕ. Now we will only extend nonlinearly the centering and whitening of the data, since after it we get standardized data in the kernel feature space $\mathcal{F}$. The iterative part of FastICA requires standardization but, although it could be done, the iterative part itself is not nonlinearized here.

Centering in $\mathcal{F}$. We shift the data $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_k)$ with its mean $E\{\phi(\mathbf{x})\}$, to obtain data

$$\begin{aligned} \phi'(\mathbf{x}_1) &= \phi(\mathbf{x}_1) - E\{\phi(\mathbf{x})\} \\ &\vdots \\ \phi'(\mathbf{x}_k) &= \phi(\mathbf{x}_k) - E\{\phi(\mathbf{x})\} \end{aligned} \tag{30}$$

with a mean of $\mathbf{0}$.

Whitening in $\mathcal{F}$. Much like that in linear ICA, the goal of this step is to find a transformation matrix Q such that the covariance matrix

$$\hat{\mathcal{C}} = \frac{1}{k}\sum_{i=1}^{k} \hat{\phi}(\mathbf{x}_i)\hat{\phi}(\mathbf{x}_i)^\top \tag{31}$$

of the sample

$$\begin{aligned} \hat{\phi}(\mathbf{x}_1) &= Q\phi'(\mathbf{x}_1) \\ &\vdots \\ \hat{\phi}(\mathbf{x}_k) &= Q\phi'(\mathbf{x}_k) \end{aligned} \tag{32}$$

is a unit matrix. As we saw earlier the column vectors of Q are the weighted eigenvectors of the positive semidefinite matrix

$$\mathcal{C} = \frac{1}{k}\sum_{i=1}^{k} \phi'(\mathbf{x}_i)\phi'(\mathbf{x}_i)^\top. \tag{33}$$

That is, based on the formula we got in Kernel-PCA (28), the transformation matrix can be defined by

$$\begin{aligned} Q &= A\bar{X}, \\ A &= [\lambda_1^{-1/2}\boldsymbol{\alpha}_1, \ldots, \lambda_m^{-1/2}\boldsymbol{\alpha}_m]^\top, \\ \bar{X} &= [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_k)], \end{aligned} \tag{34}$$

where $(\boldsymbol{\alpha}_1, \lambda_1), \ldots, (\boldsymbol{\alpha}_m, \lambda_m)$ are the dominant m eigenpairs of (25).

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the Kernel-ICA transformation can be made using $\mathbf{z}^* = WQ\phi(\mathbf{z}) = WA\kappa(X, \mathbf{z})$. Here W denotes the orthogonal transformation matrix we obtained as the output from the iterative section of FastICA, while Q is the matrix obtained from kernel centering & whitening.

Practically speaking, Kernel-FastICA = Kernel-Centering + Kernel-Whitening + iterative section of the original FastICA.

8.4. LDA in Kernel Feature Spaces

Let us consider the following function for a fixed κ, ϕ and $\mathcal{F}$.

$$\tau(\mathbf{v}) = \frac{\mathbf{v}^\top \mathcal{B}\mathbf{v}}{\mathbf{v}^\top \mathcal{W}\mathbf{v}}, \qquad \mathbf{v} \in \mathcal{F} \setminus \{\mathbf{0}\}, \tag{35}$$

where the matrices needed for LDA are now given in $\mathcal{F}$ [19]:

$$\begin{aligned} \mathcal{B} &= \textstyle\sum_{j=1}^{r} \frac{k_j}{k}(\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^\top \\ \mathcal{W} &= \textstyle\sum_{j=1}^{r} \frac{k_j}{k}\mathcal{C}_j \\ \boldsymbol{\mu} &= \textstyle\frac{1}{k}\sum_{i=1}^{k} \phi(\mathbf{x}_i) \\ \boldsymbol{\mu}_j &= \textstyle\frac{1}{k_j}\sum_{\mathcal{L}(i)=j} \phi(\mathbf{x}_i) \\ \mathcal{C}_j &= \textstyle\frac{1}{k_j}\sum_{\mathcal{L}(i)=j} (\phi(\mathbf{x}_i) - \boldsymbol{\mu}_j)(\phi(\mathbf{x}_i) - \boldsymbol{\mu}_j)^\top \end{aligned} \tag{36}$$

We may also suppose without loss of generality here that $\mathbf{v} = \sum_{i=1}^{k} \alpha_i \phi(\mathbf{x}_i)$ holds during the search for the stationary points of (35). With this assumption, after some algebraic rearrangement we obtain the formula

$$\tau(\mathbf{v}) = \frac{\mathbf{v}^\top \mathcal{B} \mathbf{v}}{\mathbf{v}^\top \mathcal{W} \mathbf{v}} = \frac{\boldsymbol{\alpha}^\top K \Theta_1 K \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top K \Theta_2 K \boldsymbol{\alpha}} = \tau(\boldsymbol{\alpha}), \tag{37}$$

where K is the kernel matrix, $\Theta_1 = I - \bar{1}$ and

$$[\Theta_2]_{ij} = \begin{cases} \frac{1}{k_t} - \frac{2}{k} + \frac{1}{k^2}, & \text{if } t = \mathcal{L}(i) = \mathcal{L}(j) \\ -\frac{2}{k} + \frac{k_t + k_s}{k^2}, & \text{if } t = \mathcal{L}(i) \neq \mathcal{L}(j) = s \end{cases} \tag{38}$$

This means that (35) can be expressed as dot products of $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_k)$ and that the stationary points of this equation can be computed using the real eigenvectors of $(K\Theta_2 K)^{-1} K \Theta_1 K$. Since in general $K\Theta_2 K$ is a positive semidefinite matrix, it can be forced to be invertible using the technique presented in the subsection of LDA. For defining the transformation matrix V of Kernel-LDA we will use only those eigenvectors which correspond to the m dominant real eigenvalues, denoted by $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m$.

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the Kernel-LDA transformation can be performed using $\mathbf{z}^* = V\bar{X}\phi(\mathbf{z}) = V\kappa(X, \mathbf{z})$.

8.5. SDA in Kernel Feature Spaces

Now let the dot product be implicitly defined by the kernel function κ in some finite or infinite dimensional feature space $\mathcal{F}$ with associated transformation ϕ:

$$\kappa(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z}). \tag{39}$$

Let $\delta(\mathbf{v})$, the potential of the spring model along the direction $\mathbf{v}$ in $\mathcal{F}$, be defined by

$$\sum_{i,j=1}^{k} ((\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^\top \mathbf{v})^2 [\Theta]_{ij}, \tag{40}$$

where

$$[\Theta]_{ij} = \begin{cases} -1, & \text{if } \mathcal{L}(i) = \mathcal{L}(j) \\ 1, & \text{otherwise} \end{cases} \qquad i, j = 1, \ldots, k. \tag{41}$$

Technically speaking, Kernel-SDA [21] searches for those directions $\mathbf{v}$ of the form $\bar{X}\boldsymbol{\alpha}$ with a variational parameter vector $\boldsymbol{\alpha}$, along which a large potential is obtained. However, instead of the function $\delta(\bar{X}\boldsymbol{\alpha})$, we use its normalized version

$$\tau(\boldsymbol{\alpha}) = \frac{\delta(\bar{X}\boldsymbol{\alpha})}{\boldsymbol{\alpha}^\top \boldsymbol{\alpha}}, \tag{42}$$

when selecting new directions. It is easy to prove that $\tau(\boldsymbol{\alpha})$ is equal to the following Rayleigh quotient formula

$$\tau(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^\top \bar{X}^\top A \bar{X} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top \boldsymbol{\alpha}}, \tag{43}$$

where

$$A = \sum_{i,j=1}^{k} (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^\top [\Theta]_{ij}. \tag{44}$$

Moreover, it is also straightforward to prove that (43) takes the following form:

$$\frac{\boldsymbol{\alpha}^\top \left(K\tilde{\Theta}K^\top - K\Theta K^\top \right) \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top \boldsymbol{\alpha}}, \tag{45}$$

where K is the kernel matrix and $\tilde{\Theta}$ is a diagonal matrix with the sum of each row of Θ in the diagonal. The stationary points of the above Rayleigh quotient formula will furnish the row vectors of the orthogonal matrix V.

After taking the derivative of (45) it is readily seen that the stationary points of $\tau(\boldsymbol{\alpha})$ can be obtained via an eigenanalysis of the following symmetric eigenproblem:

$$(K\tilde{\Theta}K^\top - K\Theta K^\top)\boldsymbol{\alpha} = \lambda \boldsymbol{\alpha}. \tag{46}$$

If we assume that the dominant m eigenvectors are $\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_m$ then the orthogonal matrix V is defined by $[\boldsymbol{\alpha}_1 c_1, \cdots, \boldsymbol{\alpha}_m c_m]^\top$, where the normalization parameter c_i is equal to $(\boldsymbol{\alpha}_i^\top K \boldsymbol{\alpha}_i)^{-1/2}$. This normalization factor ensures that the two-norm of row vectors of the transformation matrix $V\hat{X}$ is unity.

Transformation of test vectors. For an arbitrary test vector $\mathbf{z} \in \mathcal{X}$ the Kernel-SDA transformation can be done using $\mathbf{z}^* = V\bar{X}\phi(\mathbf{z}) = V\kappa(X, \mathbf{z})$.

9. Experiments

All the experiments were run on the LPCC and FBLE features described in Section 6. As mentioned previously, HMM results were obtained only for the LPCC case. Overall, the exact parameters for the learners and the transformations were as follows.

Hidden Markov modeling. In the HMM experiments 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 modeled using a mixture of four Gaussians with diagonal covariance matrices. The models were trained using the Viterbi training algorithm.

Gaussian mixture modeling. Unfortunately there is no closed formula for getting the optimal parameters of the mixture model, so the expectation-maximization (EM) algorithm is normally used to find proper parameters, but it only guarantees a locally optimal solution. This iterative technique is very sensitive to initial parameter values, so we utilized k-means clustering [23] to find a good starting parameter set. Since k-means clustering again only guaranteed finding 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 2. In all cases the covariance matrices were forced to be diagonal.

Artificial neural networks. In the ANN 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 at 150 in all experiments except in the case of LDA and KLDA where a value of 50 was found sufficient because of the enormous dimension reduction (these values were chosen empirically based on preliminary experiments). Training was stopped based on the cross-validation of 15% of the training data.

Support Vector Machine. In the experiments with SVM a third-order polynomial kernel function was applied.

As regards the transformations, in the case of (Kernel-)LDA the original 120 dimensions were reduced to just 27, the number of classes minus one. In the case of the (Kernel-)PCA and (Kernel-)ICA transformations we kept the largest m components that retained 99% of the spectrum. In our case m turned out to be 81. This was also a reasonable choice for (Kernel-)SDA. In the case of the kernel transformations we always used a Gaussian RBF kernel function.

Naturally when we applied a certain transformation on the training set before learning, we applied the same transformation on the test data during testing.

10. Results and Discussion

Table 1 shows the recognition accuracies where the columns represent the feature sets (transformed/ untransformed), while the rows correspond to the applied learning methods. For HMM we have only one score, as in this case no transformation could be applied.

Upon inspecting the results the first thing one notices is that the discriminative learners (ANN and SVM) always outperform the generative one (GMM). Hence there is a clear advantage of modeling the classes together rather than separately. Another important observation is that HMM, in spite of being a generative model, produced the highest score. But one has to keep in mind that HMM uses many more features per phoneme (the exact number depending on the segment length), and also a quite involved integration technique. Actually, we consider the fact that we could attain practically the same score with our quite simple feature extraction method as proof that the HMM technique can be easily surpassed with a more sophisticated discriminative segmental phoneme model. We should also mention here that our segmental feature calculation method was devised with FBLE preprocessing in mind and that it works much better with those features. Our current best result with FBLE is 95.55%, which shows that LPCC is definitely not an optimal choice for our system - but the goal of this paper was to compare HMM

Table I. Recognition accuracies for the phoneme classification. The maximum is in bold.

	none	**none**	**PCA**	**ICA**	**LDA**	**SDA**	**KPCA**	**KICA**	**KLDA**	**KSDA**
classifier	*variable*	120	81	81	27	81	81	81	27	81
ANN	-	90.18	90.42	88.65	89.24	90.95	90.66	89.24	89.95	91.19
GMM	-	83.33	82.47	82.09	87.44	86.70	83.21	82.44	89.83	89.36
SVM	-	90.13	89.53	88.94	90.78	90.42	90.07	89.95	**91.90**	91.48
HMM	90.66	-	-	-	-	-	-	-	-	-

and the other learners *with the same preprocessing technique.*

As regards the transformations, we found that the scores show no significant differences. However, it is clear that the supervised techniques (LDA/ KLDA and SDA/KSDA) are in general superior to the unsupervised ones (PCA/KPCA and ICA/ KICA). After LDA the learners could produce the same or similar scores in spite of the drastic dimension reduction performed (120 features reduced to 27). SDA scored very similar results, although it worked with 81 features. PCA in most cases did slightly worse. We attribute this to the fact that LPCC inherently contains an orthogonal transformation (the discrete cosine transform), so PCA could not bring any additional gain. ICA always performed the worst, which accords with our earlier findings, where we could find no real advantage of using ICA in the phoneme recognition task.

As regards the kernel-based transformations, we found that they outperform their linear versions, but not significantly. As in other tests we found them much more effective, which we attribute to the sensitivity to the parameter settings. It seems a more thorough search for their optimal parameters might have resulted in a better performance.

11. Conclusions and Future Work

The main goal of this paper was to compare several classification and transformation methods applied to phoneme classification. A further goal was to compare these scores with the performance of an HMM system. For this reason we used the LPCC feature set and a relatively simple segmental model. We found that our classifier can attain scores very similar to those of HMM, in spite of the very simple segmental modeling.

As regards the transformations, we learned that the supervised ones are the most useful. Kernel transformations proved most promising, but some of them need further experimentation. We think that it would be worth looking for other supervised techniques that could be constructed in a similar way to the SDA one.

In the future we plan to conduct more experiments to fine tune the parameters of the kernel methods, and also to examine other kernel functions as well. In the case of SDA and KSDA a more sophisticated setup of the Θ matrix might lead to a better performance. This is one of our future research aims.

As regards the applicability of the classifiers in a continuous speech recognizer, we found we could attain results just like those of HMM with the application of the segment model [30]. But the study of the effect of linear and nonlinear feature space transformations at the word level will be a subject of future work.

12. Acknowledgments

The HMM system used in our experiments [29] was trained and tested by Máté Szarvas at the Department of Telecommunications and Telematics, Technical University of Budapest. We greatly appreciate his indispensable help in making this study complete.

References

1. M. A. Aizerman, E. M. Braverman, L. I. Rozonoer, "Theoretical foundation of the potential function method in pattern recognition learning," Auttomat. Remote Cont., vol. 25, pp. 821-837, 1964.
2. F. R. Bach, and M. I. Jordan, "Kernel Independent Component Analysis," J. Machine Learning Res. vol. 3, pp. 1-48, 2002.
3. G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," Neural Comput., vol. 12, pp. 2385-2404, 2000.
4. C. M. Bishop, Neural Networks for Pattern Recognition, Oxford Univerisity Press Inc., New York, 1996.
5. B. E. Boser, I. M. Guyon, V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in Proc. of the Fifth Annual ACM Conference on Computational Learning Theory, D. Haussler (eds.), ACM Press, Pittsburg, pp. 144-152, 1992.
6. N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000.
7. P. Comon, Independent component analysis, A new concept? Signal Processing, vol. 36, pp. 287-314, 1994.
8. F. Cucker and S. Smale, "On the mathematical foundations of learning," Bull. Am. Math. Soc., vol. 39, pp. 1-49, 2002.
9. R. Duda and P. Hart, Pattern Classification and Scene Analysis, Wiley and Sons, New York, 1973.
10. FastICA Web Page, http:// www.cis.hut.fi/projects /ica/fastica/index.shtml.
11. K. Fukunaga, Statistical Pattern Recognition, Academic Press, New York, 1989.
12. M. G. Genton, "Classes of Kernels for Machine Learning: A Statistics Perspective," J. Machine Learning Res., vol. 2., pp. 299-312, 2001.
13. J. Glass, J. Chang, M. McCandless, "A probabilistic framework for feature-based speech recognition," Proceedings of ICSLP, Philadelphia, pp. 2277-2280, 1996.
14. A. Hyvärinen, "A family of fixed-point algorithms for independent component analysis," Proceedings of ICASSP, Munich, Germany, 1997.
15. A. Hyvärinen, "New Approximations of Differential Entropy for Independent Component Analysis and Projection Pursuit," In Advances in Neural Information Processing Systems, vol. 10, pp. 273-279, MIT Press, 1998.
16. I. J. Jolliffe, Principal Component Analysis, Springer-Verlag, New York, 1986.
17. Kernel Machines Web site, http://kernel-machines.org.
18. A. Kocsor, L. Tóth, A. Kuba Jr., K. Kovács, M. Jelasity, T. Gyimóthy, J. Csirik, "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.
19. A. Kocsor, L. Tóth, D. Paczolay, "A Nonlinearized Discriminant Analysis and its Application to Speech Impediment Therapy," in V. Matousek et al. (eds.): Proc. of Text, Speech and Dialogue 2001, Springer Verlag LNAI Series, vol. 2166, pp. 249-257, 2001.
20. A. Kocsor and J. Csirik, "Fast Independent Component Analysis in Kernel Feature Spaces," in: L. Pacholski and P. Ruzicka (eds.): Proc. of SOFSEM 2001, Springer Verlag LNCS Series, vol. 2234, pp. 271-281, 2001.
21. A. Kocsor and K. Kovács, "Kernel Springy Discriminant Analysis and Its Application to a Phonological Awareness teaching System," in P. Sojka et al.(eds.): Proc. of Text, Speech and Dialogue 2002, Springer Verlag LNAI Series, vol. 2448, pp. 325-328, 2002.
22. S. Mika, G. Rätsch, J. Weston, B. Schölkopf, K.-R. Müller, "Fisher discriminant analysis with kernels," in: Y.-H. Hu et al. (eds.): Neural Networks for Signal Processing IX, pp. 41-48. IEEE, 1999.
23. L. Rabiner and B.-H. Juang, Fundamentals of Speech Recognition, Prentice Hall, 1993.
24. M. Ostendorf, V. Digalakis, O. Kimball, "From HMMs to segment models: a unified view of stochastic modeling for speech recognition," IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 4, pp. 360-378, 1996.
25. R. Rosipal, L. J. Trejo, "Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Space," J. Machine Learning Res. vol. 2, pp. 97-123, 2001.
26. B. Schölkopf, A. J. Smola, K.-R. Müller, Kernel principal component analysis in Advances in Kernel Methods - Support Vector Learning, B. Schölkopf et al. (eds.), MIT Press, Cambridge, pp. 327-352, 1999.
27. B. Schölkopf, P. L. Bartlett, A. J. Smola, R. Williamson, Shrinking the tube: a new support vector regression algorithm in Advances in Neural Information Processing Systems 11, M. S. Kearns et al. (eds.), MIT Press, Cambridge, pp. 330-336, 1999.
28. B. Schölkopf and A. J. Smola, Learning with Kernels, MIT Press, Cambridge, 2002.
29. M. Szarvas, P. Mihajlik, T. Fegyó, P. Tatai, "Automatic Recognition of Hungarian: Theory and Practice," Int. Journal of Speech Technology, vol 3., no. 3/4, pp. 237-252, 2000.
30. L. Tóth, A. Kocsor, K. Kovács, "A Discriminative Segmental Speech Model and Its Application to Hungarian Number Recognition," In P. Sojka et al.(eds.):Text, Speech and Dialogue 2000, Springer Verlag LNAI series, vol. 1902, pp. 307-313, 2000.
31. V. N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
32. V. N. Vapnik, Statistical Learning Theory, John Wiley & Sons Inc., 1998.