# Kernel Discriminative Analysis for Speech Recognition

*Shankar Kumar and Peng Xu*

Center for Language and Speech Processing,
Department of Electrical and Computer Engineering,
Johns Hopkins University, Baltimore, MD 21218
skumar@clsp.jhu.edu, xp@clsp.jhu.edu

## Abstract

Linear Discriminative Analysis techniques have been used in pattern recognition to map feature vectors to achieve optimal classification. Kernel Discriminative Analysis(KDA) seeks to introduce non-linearity in this approach by mapping the features to a non-linear space before applying LDA analysis. The formulation is expressed as an eigenvalue problem resolution. Using a different kernel, one can cover a wide class of nonlinearities. In this paper, we describe this technique and present an application to a speech recognition problem. We give classification results for a connected digit recognition task and analyze some existing problems.

## 1. Introduction

In classification and other data analysis tasks, it is often necessary to utilize pre-processing on the data before applying the algorithm at hand and it is common to first extract features suitable for the tasks to solve. Linear discriminant analysis (LDA) is a traditional statistical method which has proven successful for classification problems [2]. The procedure is based on an eigenvalue resolution and gives an exact solution of the maximum of the inertia. LDA addresses the following question: Given a data set with two classes, which is the best feature or feature set to discriminate the two classes? Starting from the optimal Bayes classifier and by assuming Gaussian distributions for the classes, the closed form solution of LDA can be derived. Apart from having a closed form solution, LDA is also well known for being robust against noise. But this method fails for a nonlinear problem. Extending LDA with the kernel idea, we can find the best discriminant feature or feature set in a more general case (either linear or nonlinear). The kernel idea was originally applied in Support Vector Machines [9], kernel PCA and other kernel based algorithms to define a nonlinear mapping from the data space to some feature space. Starting from this work, we show how to express our approach as a linear algebraic formula in the transformed space using the kernel idea. The references [3] and [4] have different derivations for KDA based on two class problems. We extended their approaches to a general $l$-class formulation.

Hidden Markov Models (HMMs) are widely used in speech recognition systems. An HMM contains a finite number of states connected by transitions. Speech recognition in the HMM framework can also be treated as a classification problem where each state of an HMM forms a class. Once we have labeled data, we can apply LDA or any other discriminant analysis just as any other classification problem.

In the rest of the paper, we first briefly review Fisher's linear discriminant and formulation of LDA in section 2, then extend LDA to Kernel Discriminant Analysis (KDA) in section 3. In section 4, we describe our application of KDA to a speech recognition system. Some experimental results will be shown in section 5. And finally, we present our conclusions in section 6.

## 2. Fisher's Linear Discriminant Review

Let $\mathcal{X}_1 = x_1^1, ..., x_{n_1}^1$ and $\mathcal{X}_2 = x_1^2, ..., x_{n_2}^2$ be samples from two different classes and $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 = x_1, ..., x_n$ be all the samples. Fisher's linear discriminant is given by the vector which maximizes $J(w) = \frac{w^T S_B w}{w^T S_W w}$, that is,

$$\hat{w} = argmax_w J(w) = argmax_w \frac{w^T S_B w}{w^T S_W w}$$

where

$$S_B = (m_1 - m_2)(m_1 - m_2)^T$$

$$S_W = \sum_i \sum_{x \in \mathcal{X}_i} (x - m_i)(x - m_i)^T$$

are the between and within class scatter matrices respectively and $m_i$ is defined by $m_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_j^i$. The intuition behind maximizing $J(w)$ is to find a direction which maximizes the projected class means while minimizes the classes variances in this direction.

The solution to this problem is to find the eigenvector of $S_W^{-1} S_B$ which corresponds to the largest eigenvalue of the same matrix. Or equivalently, to find solutions for the equation:

$$\lambda S_W w = S_B w$$

and then choose $\hat{w}$ corresponding to the largest value of $\lambda$.

If we assume Gaussian distributions for all classes and assume equal covariance structure for all classes, we know that this solution is the same as the discriminant in the corresponding Bayes optimal classifier. The optimal Bayes classifier compares the a posteriori probabilities of

all classes and assigns a pattern to the class with the maximal probability. Although LDA relies on heavy assumptions which are not true in many applications, it has proven very powerful not only because of the simple closed form solution, but also the robustness of a linear model.

## 3. Kernel Discriminative Analysis (KDA)

For most of the real world applications (e.g., speech recognition), simple LDA is not complex enough. We also have much more classes than just two and there are very different number of date points in each class. To increase the discriminant power of the analysis, we could find nonlinear directions by first mapping the data nonlinearly into some feature space $\mathcal{F}$ and computing LDA there. Thus we will implicitly yield a nonlinear discriminant in input space.

### 3.1. Formulation of KDA

We will derive formulas for KDA assuming we have $l$ classes and each class has $n_i$ data samples. We also assume that the total number of data samples is $n$. So we have $\mathcal{X}_i = x_1^i, ..., x_{n_i}^i$ for $i = 1, 2, ..., l$.

Let $\Phi$ be a nonlinear mapping from original data samples to some feature space $\mathcal{F}$. Then to find the linear discriminant in space $\mathcal{F}$, we need to maximize

$$J(w) = \frac{w^T S_B^\Phi w}{w^T S_W^\Phi w}$$

where $w \in \mathcal{F}$, $S_B^\Phi w$ and $S_W^\Phi w$ are the corresponding matrices in $\mathcal{F}$. In our $l$ classes case, we have

$$S_B^\Phi = \sum_{i=1}^l \frac{n_i}{n}(m_i^\Phi - m^\Phi)(m_i^\Phi - m^\Phi)^T$$

$$S_W^\Phi = \sum_{i=1}^l \frac{n_i}{n}(\frac{1}{n_i}\sum_{j=1}^{n_i}(\Phi(x_j^i) - m_i^\Phi)(\Phi(x_j^i) - m_i^\Phi)^T)$$

where

$$m_i^\Phi = \frac{1}{n_i}\sum_{j=1}^{n_i}\Phi(x_j^i)$$

$$m^\Phi = \frac{1}{l}\sum_{i=1}^l \frac{n_i}{n} m_i^\Phi = \frac{1}{n}\sum_{j=1}^n \Phi(x_j)$$

and we have some abuse of notation to let $\mathcal{X} = \mathcal{X}_1 \cup ... \cup \mathcal{X}_l = x_1, ...x_n$.

Since directly compute $\Phi(x)$ is always not feasible, we can introduce a kernel function $k(x, y) = (\Phi(x) \cdot \Phi(y))$ which is a dot product in the feature space $\mathcal{F}$. So instead of mapping the data explicitly we use only dot product of the training patterns. Possible choice of $k$ includes Gaussian RBF and polynomial kernels and we will discuss this later.

From the theory of reproducing kernels we know that any solution $w \in \mathcal{F}$ must lie in the span of all training samples in $\mathcal{F}$. Therefore we can find an expansion for $w$ of the form

$$w = \sum_{j=1}^n \alpha_j \Phi(x_j)$$

Using this expansion and the definitions above, after some mathematical manipulations, we will get:

$$w^T S_B w = \boldsymbol{\alpha}^T M \boldsymbol{\alpha}$$

$$w^T S_W w = \boldsymbol{\alpha}^T N \boldsymbol{\alpha}$$

where

$$M = \sum_{i=1}^l \frac{n_i}{n}(M_i - M_0)(M_i - M_0)^T$$

$$(M_i)_j = \frac{1}{n_i}\sum_{k=1}^{n_i} k(x_j, x_k^i), j = 1 \sim n$$

$$(M_0)_j = \frac{1}{n}\sum_{k=1}^{n} k(x_j, x_k), j = 1 \sim n$$

$$N = \sum_{i=1}^l \frac{1}{n} K_i (I_{n_i} - 1_{n_i}) K_i^T$$

$$(K_i)_{pq} = k(x_p, x_q^i), p = 1 \sim n, q = 1 \sim n_i$$

and $I_{n_i}$ is identity matrix of size $n_i \times n_i$, $1_{n_i}$ is a matrix of size $n_i \times n_i$ with all entries equal to $\frac{1}{n_i}$, $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n)$.

Combining all the equations above, we can find Fisher's discriminant in $\mathcal{F}$ by maximizing

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T M \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T N \boldsymbol{\alpha}}$$

whose solution is analogous to finding the leading eigenvector of $N^{-1}M$. If we want to have $r$ dimensional vectors in space $\mathcal{F}$, we can find the $r$ leading eigenvectors of $N^{-1}M$ and put them into a matrix $R$. Each $\boldsymbol{\alpha}$ is a row in $R$. Each projection of a new data sample $x$ to $w$ is given by

$$(w \cdot \Phi(x)) = \sum_{j=1}^n \alpha_j k(x_j, x)$$

### 3.2. Practical Considerations

In the above formulations, $N$ and $M$ are both of size $n \times n$, which in practice can be very big. Also, $N$ is obvious a singular matrix because we are estimating $n$ dimensional covariance structure from $n$ samples. In order to make $N$ a positive matrix and at the same time preserve its eigenvalues, we can simply add a small constant $\mu$ to its diagonal components, i.e., we use $N_\mu = N + \mu I$ to replace $N$.

As mentioned in 3.1, some kernel functions have proven useful, e.g. Gaussian RBF $k(x, y) = exp(- \parallel x - y \parallel^2 / c)$, or polynomial kernels $k(x, y) = (a + b \times x \cdot y)^d$, for some constant a, b, c, and d. Usually those constants are chosen for the best performance. In this paper, we give results on only polynomial kernels for simplicity. But we do not make the assumption that polynomial kernels work the best for our problem.

## 4. Application to Speech Recognition

### 4.1. Hidden Markov Models(HMM)

A Hidden Markov Models(HMM) [8] can be formally described as follows:

1. A state space $\mathcal{X} = \{x_{(1)}, x_{(2)}, .., x_{(P)}\}$; the hidden random variables $X_k$ take values in $\mathcal{X}$.

2. An observation space $\mathcal{Y}$; the observed random variables $Y_k$ take values in $\mathcal{Y}$. For simplicity, we assume that this space is discrete and given by $\mathcal{Y} = \{y_{(1)}, y_{(2)}, .., y_{(Q)}\}$.

3. Transition probability distributions
   $P(X_{k+1} = x_{(i)} | X_k = x_{(j)})$.
4. Output probability mass function
   $P(Y_k = y_{(i)} | X_k = x_{(j)})$
5. Initial state distribution
   $\pi = \{\pi_i\};\ \pi_i = P(X_1 = x_{(i)});\ 1 \leq i \leq N$.

In continuous density HMMs, the output alphabet $Y \in \mathcal{R}^d$ where $d$ is the dimensionality of the input features and the probability mass function is replaced by probability density function (p.d.f). This density function is usually a mixture of Gaussian densities with mean vectors $\boldsymbol{\mu}$ and Covariance matrices $\boldsymbol{\Sigma}$.

### 4.2. A Simple Speech Recognition System

Our speech recognition experiments were based upon a HMM based speech recognizer for TIDIGITS connected digits task [7]. The task vocabulary is made of the digits '1' to '9', plus 'oh' and 'zero', for a total of 11 words. Each speaker contributes to the corpus with two repetitions of each digit in isolation and 55 digit strings, evenly distributed into lengths 2,3,4,5 and 7. There are a total of 326 speakers. We chose 87 digit strings for our training set and 870 utterances for our test set.

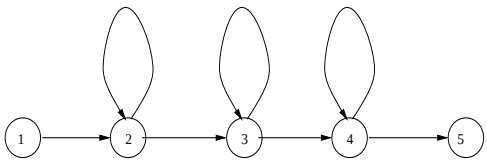


Figure 1: Diagram of a HMM with 3 emitting states numbered 2, 3 and 4

The baseline HMM system built with HTK Toolkit [6] consists of 11 word models, and a model for silence. Each word model had 8 emitting states while the silence model had 3 emitting states, summing to a total of 91 states. The diagram of HMM with 3 emitting states is given in Figure 1. The front-end features of this baseline system were 12 Mel-frequency cepstral coefficients plus a energy term and their deltas and acceleration terms, leading to a 39 dimensional vector for each speech frame (We have 100 frames per second). The observation densities on the HMM states were single mixture Gaussian densities. The parameters of this baseline system were estimated using standard Expectation-Maximization (Baum-Welch) procedures and Word Error Rate(WER) of the system was 0.69% on the training set and 5.17% on the test set.

We use the baseline HMMs to perform Viterbi alignment over the training data. This procedure finds the optimal state sequence $x$ given the observation sequence $y$ and the word sequence $w$ given the model(HMM).

$$x^{opt} = \text{argmax}_{x \in \mathcal{X}^n}\ P(x|w, y).$$

The forced alignment tags each observation $y_i$ in the training data with its state label $x_i$.

### 4.3. Applying KDA: Procedure and Implementation

To apply KDA to our speech recognition system, we need labeled training data. As described in 4.1 and 4.2, the labels come from Viterbi alignment of the training data. The overall experimental procedure is as following:

1. Estimate HMM parameters in the original data space using Baum-Welch algorithm and get baseline result
2. Perform forced Viterbi alignment over training data to get class labels, i.e., the HMM state sequence
3. Perform KDA to obtain the $r$ leading eigenvectors of the matrix $N^{-1}M$
4. Transform the original data into space $\mathcal{F}$
5. Re-estimate HMM parameters in $\mathcal{F}$ and get decoding results

Note here we will repeat step 3-5 for every kernel function.

There are some implementation issues when we follow the above procedure with our speech recognition system. We have about 13,000 data samples in the training data, which means the size of matrices $N$ and $M$ is about 13,000$\times$13,000. To invert such a huge matrix and to do the multiplication are practically very time consuming. So in order to really implement our KDA, we need to sample the training data to reduce the size of $N$ and $M$ to about 2,000$\sim$4,000. Our sampling scheme is such that the within class variances are approximately maximized while the within class means are approximately the same as before sampling. Of course there are many ways of sampling the data, but we assume this particular scheme will work for our purpose at this time. As we mentioned in 3.2, we use only polynomial kernels in our experiments, i.e., $k(x, y) = (a + b \times x \cdot y)^d$. In our experiments, we fixed $d = 3$ and tried some different numbers for $a$ and $b$. We chose $a$ and $b$ in the range such that all entries in the kernel matrix are positive and in some appropriate interval, say, [0,1] or [1,2]. This is only a very preliminary study of kernel functions and we expect to explore this problem more in the future.

### 4.4. HMM Re-estimation Schemes

The MFCC feature vectors on the training and test sets are transformed by the KDA procedure described in the section 4.3. To evaluate the effect of these new features, we need to re-estimate the parameters of the baseline HMM. This re-estimation can be done through a variety of ways. We explored 3 different schemes which can be described as follows:

1. Re-estimation I (R-I): In this procedure, we train HMMs from scratch on transformed feature vectors. This training is standard EM(Baum-Welch) estimation and similar to the training procedure of our Baseline HMMs.

2. Re-estimation II-A (R-IIA): This procedure consists of re-estimating means and variances of HMMs on transformed feature vectors and termed as Single-Pass retraining(SPR) in the literature [5]. Given a HMM system trained on our baseline MFCC features, SPR assumes that the frame/state alignment is identical for the MFCC and the KDA-transformed features. SPR first computes the a posteriori probability of state occupation using the Baseline models and the MFCC vectors. Using this alignment, the Gaussian parameters are updated using the corresponding observations from the transformed data.

3. Re-estimation II-B (R-IIB): This is a refinement of scheme R-IIA by performing further training (Baum-Welch) iterations on the resulting models by updating all parameters (Means,Variances and Transitions Probabilities) on the transformed features.

## 5. Results and Analysis

In this section, we describe the word error rate (WER) results obtained by decoding procedure using different re-estimation schemes described in 4.4, different training sample sizes for KDA and different dimensions of transformed feature vectors. Table 1 shows the results from Re-estimation Scheme R-I.

| Dimension | 39 | 20 | 12 |
|---|---|---|---|
| Train(2000) | 3.48 | 1.38 | 1.04 |
| Test(2000) | 24.84 | 14.26 | 11.57 |
| Train(4000) | 1.04 | 1.04 | 1.73 |
| Test(4000) | 15.44 | 11.08 | 10.42 |

Table 1: WER Scheme R-I: Varying Sample Size and Dimension

In Table 1, we get best results on test set by using 12 dimensional features. We also note that increasing training sample size improves performance considerably which implies that we can get even better results by training KDA on all samples.

Table 2 shows the results from Re-estimation Scheme R-IIA and R-IIB. In Table 3, we compare WER results of R-I and R-II with the baseline. The transformed data of R-II is 39 dimensional. From Table 2 , we can still observe that increasing the training sample size improves the performance. Using this scheme, we can start re-training the HMM from the same point that we train the classifier. This leads to a much better classifier performance relative to R-I on the training set as seen in Table 3. But, we note that polynomial kernel fails to generalize well on the test set. This is possible because the training set for KDA is quite small. Re-estimation Scheme R-II could not be carried out for different dimensions due to limitations of the toolkit. So, the effect of varying feature dimensions could not be investigated.

| Sample Size | 2000 | 4000 |
|---|---|---|
| Train(R-IIA) | 0.35 | 0.35 |
| Test(R-IIA) | 19.53 | 18.48 |
| Train(R-IIB) | 0 | 0 |
| Test(R-IIB) | 17.71 | 16.35 |

Table 2: WER Schemes R-II: Varying the Sample Size

| Schemes | Baseline | KDA |
|---|---|---|
| Train(R-I) | 0.69 | 1.04 |
| Test(R-I) | 5.17 | 11.57 |
| Train(R-IIB) | 0.69 | 0 |
| Test(R-IIB) | 5.17 | 17.71 |

Table 3: WER Comparison to Baseline for Schemes R-I and R-II

## 6. Conclusions and Future Work

On training data, we observe that KDA performs distinctly better than the baseline. But, the procedure is not computationally feasible for a LVCSR system due to enormous sizes of the matrices involved. As future work, there may be other class definitions (e.g. phonemes, words etc.) and incremental approaches to estimating KDA that may make the approach feasible. This may also improve the generalization ability of the procedure.

## 7. References


[1] R. O. Duda and P. E. Hart , “Pattern classification and scene analysis”, Wiley Publishers, New York, NY, 1973.

[2] K. Fukunaga, “Introduction to Statistical Pattern Recognition”, Second Edition, Academic Press, 1990.

[3] S. Mika and G. Rätsch and J. Weston and B. Schölkopf and K.-R. Müller, “Fisher Discriminant Analysis with Kernels”, Neural Networks for Signal Processing IX, 1999.

[4] G. Baudat and F. Anouar, “Generalized Discriminant Analysis Using a Kernel Approach”, Neural Computation, Vol. 12, No. 1, 2000.

[5] P. C. Woodland et. al. The HTK Large Vocabulary Recognition Sysyem for the ARPA H3 task, Proc. DARPA Speech Recognition Workshop, 1996.

[6] S. Young et.al, The HTK Book, Version 3.0, July 2000.

[7] R. G. Leonard, “A database for speaker-independent digit recognition”, Proc. of ICASSP-84, 1984.

[8] F. Jelinek, “Statistical Methods for Speech Recognition”, MIT Press, 1997.

[9] V. Vapnik, “The Nature of Statistical Learning Theory”, Springer, New York, N.Y., 1995.