Linear Feature Transformations in Slovak Phoneme-Based Continuous Speech Recognition

Jozef Juhár and Peter Viszlay

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/48715

1. Introduction

The most common acoustic front-ends in automatic speech recognition (ASR) systems are based on the state-of-the-art Mel-Frequency Cepstral Coefficients (MFCCs). The practice shows that this general technique is good choice to obtain satisfactory speech representation. In the past few decades, the researchers have made a great effort in order to develop and apply such techniques, which may improve the recognition performance of the conventional MFCCs. In general, these methods were taken from mathematics and applied in many research areas such as face and speech recognition, high-dimensional data and signal processing, video and image coding and many other. One group of mentioned methods is represented by linear transformations.

Linear feature transformations (also referred as subspace learning or dimensionality reduction methods) are used to convert the original data set to an alternative and more compact set with retaining of information as much as possible. They are also used to increase the robustness and the performance of the system. In speech recognition, the basic acoustic front-end based on MFCCs can be supplemented by some kind of linear feature transformation. The linear transformation is applied in feature extraction step. Then the whole feature extraction process is achieved in two steps: parameter extraction and feature transformation. Linear transformation is applied to a sequence of acoustic vectors obtained by some kind of preprocessing method. Usually, the spectral, log-spectral, Mel-filtered spectral or cepstral features are projected to a more relevant and more decorrelated subspace, which is directly used in acoustic modeling. During the transformation often a dimension reduction step is also done. This is achieved by retaining only the relevant dimensions after the transformation according to some optimization criterion. The dimension reduction step helps to solve the problem called the curse of dimensionality.

In practice, supervised and unsupervised subspace learning methods are used. The most popular data-driven unsupervised transformation used in ASR is Principal Component Analysis (PCA). It is known that the supervised methods need an information about the structure of the data, which are partitioned in the classes. Therefore, it is necessary to use appropriate class labels. A widely used supervised method is known as Linear Discriminant Analysis (LDA).

In numerous research works and publications it was proven that the above mentioned linear transformations were successfully applied in ASR to multiple languages with different characteristics of speech. The Slovak speech recognition research group tends to follow this trend. In this work, we present a practical methodology with adequate theoretical principles related to application of linear feature transformations in Slovak phoneme-based large vocabulary continuous speech recognition (LVCSR).

The main subject of this chapter is the application of LDA in Slovak ASR, but the core of most experiments is based on Two-dimensional LDA (2DLDA), which is an extension of LDA. Several context lengths of basic vectors are used in the discriminat analysis and different final dimensions of transformation matrix are utilized. The classical procedures by several our modifications are supplemented. The second part of the chapter is oriented to PCA and to our proposed method related to PCA training from limited amount of training data. The third part investigates the interaction of the above mentioned PCA and 2DLDA applied in one recognition task. The closing part compares and evaluates all experiments and concludes the chapter by presenting the best achieved results.

This chapter is divided into few basic units. Sections 2 and 3 describe LDA and 2DLDA used in speech recognition. Section 4 surveys PCA and also presents the proposed partial-data trained PCA method. Section 5 presents the setup of the system for continuous phoneme-based speech recognition. Section 6 presents extensive experiments and evaluations of the used methods in different configurations. Finally Section 7 concludes the chapter. Section 8 gives the future intentions in our research.

2. Conventional Linear Discriminant Analysis (LDA)

Linear discriminant analysis is a well-known dimensionality reduction and transformation method that maps the N-dimensional input data to p-dimensional ($p < N$) subspace while retaining maximum discrimination information. A general mathematical model of linear transformation can be written in the following manner:

$$\mathbf{y} = W^T \mathbf{x}, \tag{1}$$

where $\mathbf{y}$ is the output transformed feature set, W is the transformation matrix and $\mathbf{x}$ is the input feature set. The aim of LDA is to find this transformation matrix W with respect to some optimization criterion (information loss, class discrimination, ...). It can be obtained by applying an eigendecomposition to the covariance matrices. The p best functions resulted from the decomposition are used to transform the feature vectors to reduced representation.

2.1. Mathematical background

According to [1, 7, 11, 14, 19] LDA can be defined as follows. Suppose a training data matrix $X \in \Re^{N \times n}$ with n column vectors $\mathbf{x}_i$, where $1 \leq i \leq n$. LDA finds a linear transformation represented by transformation matrix $W \in \Re^{N \times p}$ that maps each column $\mathbf{x}_i$ of X to a column vector $\mathbf{y}_i$ in the p-dimensional space as:

$$\mathbf{y}_i = W^T \mathbf{x}_i;\ p < N. \tag{2}$$

Consider that the original data is partitioned into k classes as $X = \{\Pi_1, \ldots, \Pi_k\}$, where the class Π_i contains n_i elements (feature vectors) from the ith class. Notice that $n = \sum_{i=1}^{k} n_i$. The classes can be represented by *class mean vectors*

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{x \in \Pi_i} \mathbf{x} \tag{3}$$

and their *class covariance matrices*

$$\Sigma_i = \sum_{x \in \Pi_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T, \tag{4}$$

which are defined to quantify the quality of the cluster. Since LDA in ASR mostly in class-independent manner is used, we define the *within-class covariance matrix* as the sum of all class covariance matrices

$$\Sigma_W = \frac{1}{n} \sum_{i=1}^{k} \Sigma_i = \frac{1}{n} \sum_{i=1}^{k} \sum_{x \in \Pi_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T. \tag{5}$$

To quantify the covariance between classes, the *between-class covariance matrix* is used. It is defined as:

$$\Sigma_B = \frac{1}{n} \sum_{i=1}^{k} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T, \tag{6}$$

where

$$\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^{k} \sum_{x \in \Pi_i} \mathbf{x} \tag{7}$$

is the *global mean vector* (computed disregarding the class label information). Note that the variable $\mathbf{x}$ in speech recognition represents a *supervector* created by concatenating of acoustic vectors computed on successive speech frames. To build a supervector of J acoustic vectors (J is typically 3, 5, 7, 9 or 11 frames), the vector $\mathbf{x}_j$ at the current position j is spliced together with $\frac{J-1}{2}$ vectors on the left and right as

$$\mathbf{x} = \left[\mathbf{x}[j - \tfrac{J-1}{2}] \ \ldots \ \mathbf{x}[j] \ \ldots \ \mathbf{x}[j + \tfrac{J-1}{2}] \right]. \tag{8}$$

It should be noted that in case, when the length of the supervector was greater than the number of classes ($13 \times J > k$, where $J \geq 5, k = 45$), the between-class covariance matrix became close to singular or singular. This fact resulted in eigendecomposition with complex valued transformation matrix, which was undesirable.

Therefore, we used for these cases a modified computation of Σ_B according to [7] as follows:

$$\widetilde{\Sigma}_B = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T. \tag{9}$$

This way of computation can be interpreted as a finer estimation of Σ_B because each training supervector contributes to a final estimation of Σ_B (more data points are used) in comparison with the estimation represented by Equation 6.

The given covariance matrices are used to formulate the optimization criterion for LDA, which tries to maximize the between-class scatter (covariance) over the within-class scatter (covariance). It can be shown that the covariance matrices resulting from the linear transformation W (in the p-dimensional space) become $\widetilde{\Sigma}_B^p = W^T\Sigma_B W$ and $\widetilde{\Sigma}_W^p = W^T\Sigma_W W$. The objective function can be defined as

$$J(W) = \frac{|\widetilde{\Sigma}_B|}{|\widetilde{\Sigma}_W|} = \frac{|W^T\Sigma_B W|}{|W^T\Sigma_W W|}. \tag{10}$$

This optimization problem is equivalent to the generalized eigenvalue problem

$$\Sigma_B\mathbf{v} = \lambda\Sigma_W\mathbf{v}, \text{ for } \lambda \neq 0, \tag{11}$$

where $\mathbf{v}$ is a square matrix of eigenvectors and λ represents the eigenvalues. The solution can be obtained by applying an eigendecomposition to the matrix

$$\Sigma_W^{-1}\Sigma_B. \tag{12}$$

The reduced representation W_p of W is made by choosing p eigenvectors corresponding to p largest eigenvalues.

2.2. Class definition in LDA

Since LDA is a supervised method, it needs additional information about the class structure of the training data. In the past few years, several choices for LDA class definition in ASR were proposed and experimentally investigated. For small vocabulary phoneme-based ASR systems LDA yielded an improvement with phone level conventional class definition [4, 8]. In these cases the Viterbi-trained context independent phonemes are used as classes. For HMM-based recognizers the time-aligned HMM states can define the classes [14]. Another reasonable method is to use the subphone levels as LDA classes [15]. We showed in our work [17] that an alternative phonetic class definition based on phonetic segmentation can lead to improvement.

For large vocabulary phoneme-based ASR systems there exist several ways to define the classes. One might argue that the conventional phone-level definition is the appropriate one. For triphone-based recognizers the context-dependent or context-independent triphones can be used [13] or the tied states in context dependent acoustic models [6].

In this work we used the conventional phone-level classes for LDA and 2DLDA. The phonetic segmentation was obtained from embedded training and automatic phone alignment (see

Section 5.3). Thus, the number of classes in LDA-based experiments was identical with the number of phonemes and also with the number of trained monophone models. The disadvantage of the phone segmentation obtained from embedded training can be potentially the inaccuracy of the determined phone boundaries compared to the actual boundaries.

3. Two-Dimensional Linear Discriminant Analysis

Linear Discriminant Analysis used as a feature extraction or dimension reduction method in applications with high-dimensional data may not perform always optimally. Especially, when the dimension of the data exceeds the number of data points, the scatter matrices can become singular. This is known as the singularity or undersampled problem in LDA, which is its intrinsic limitation.

Two-Dimensional Linear Discriminant Analysis (hereinafter 2DLDA) [19] was primarily designed to overcome the singularity problem in classical LDA. 2DLDA overcomes the singularity problem implicitly. The key difference between LDA and 2DLDA is in the data representation model. While conventional LDA works with vectorized representation of data, the 2DLDA algorithm works with data in matrix representation. Therefore, the data collection is performed as a collection of matrices, instead of a single large data matrix. This concept has been used for example in [18] for PCA.

It is known that the optimal transformation matrix in LDA can be obtained by applying an eigendecomposition to the scatter matrices. Generally, these matrices can be singular because they are estimated from high-dimensional data. In recent years, several approaches have been developed to solve such problems related to high-dimensional computing [10]. One of these approaches is called PCA+LDA and it is a widely used two-stage algorithm especially in face recognition [3]. All mentioned methods require the computation of eigendecomposition of large matrices, which can lead to degradation of the efficiency.

2DLDA alleviates the difficult computation of the eigendecomposition in methods discussed above. Since it works with matrices instead of high-dimensional supervectors (as in classical LDA), the eigendecomposition in 2DLDA is computed on matrices with much smaller sizes than in LDA. This reduces the processing time and memory costs of 2DLDA compared to LDA.

3.1. Mathematical description

Let $A_i \in \mathbf{R}^{r\times c}$, $\langle 1; n\rangle$ be the n training speech signals in the corpus. Suppose there are k classes $\Pi_1, \ldots, \Pi_k$, where Π_i has n_i feature vectors. Let

$$M_i = \frac{1}{n_i} \sum_{X\in\Pi_i} X, \quad i \in \langle 1; k\rangle \tag{13}$$

be the mean of the i-th class and

$$M = \frac{1}{n} \sum_{i=1}^{k} \sum_{X\in\Pi_i} X \tag{14}$$

be the global mean. In [19], for face recognition, X originally represents a training image. For speech recognition, X represents the concatenated acoustic vectors (supervector) computed on successive speech frames [12]. In fact, X is a matrix composed by combination of acoustic vectors computed on successive speech frames. We can call this matrix analogously to supervector as supermatrix.

2DLDA considers an $(l_1 \times l_2)$-dimensional space $\mathcal{L} \otimes \mathcal{R}$, which is a tensor product of the spaces - $\mathcal{L}$ spanned by vectors $\{u_i\}_{i=1}^{l_1}$ and $\mathcal{R}$ spanned by vectors $\{v_i\}_{i=1}^{l_2}$. Since in 2DLDA, the speech is considered as a two-dimensional element, two transformation matrices, L and R are defined as $L = [u_1, \ldots, u_{l_1}], L \in \mathbf{R}^{r \times l_1}$ and matrix $R = [v_1, \ldots, v_{l_2}], R \in \mathbf{R}^{c \times l_2}$. These matrices map each $A_i \in \mathbf{R}^{r \times c}$ to a matrix $B_i \in \mathbf{R}^{l_1 \times l_2}$ as:

$$B_i = L^T A_i R, \quad i \in \langle 1; n \rangle. \tag{15}$$

Due to difficult computing of optimal L and R simultaneously, [19] derived an iterative algorithm, which for fixed R computes the optimal L. With computed L it can be updated R. The procedure is several times repeated. As in classical LDA, the scatter matrices are computed similarly, but in two-dimensional concept. Note that in 2DLDA are defined two within-class scatter matrices S_w^R and S_w^L and two between-class scatter matrices S_b^R and S_b^L concurrently. Scatter matrices coupled with R are defined as follows:

$$S_w^R = \sum_{i=1}^{k} \sum_{X \in \Pi_i} (X - M_i) R R^T (X - M_i)^T, \tag{16}$$

$$S_b^R = \sum_{i=1}^{k} n_i (M_i - M) R R^T (M_i - M)^T. \tag{17}$$

For fixed R, L can be then computed by solving an optimization problem:

$$max_L \; trace\Big(\big(L^T S_w^R L\big)^{-1} \big(L^T S_b^R L\big) \Big). \tag{18}$$

This problem can be solved as an eigenvalue problem:

$$S_w^R \mathbf{x} = \lambda S_b^R \mathbf{x}. \tag{19}$$

L can be then obtained in similar way as in LDA by applying an eigendecomposition to matrix resulting from:

$$\big(S_w^R\big)^{-1} S_b^R. \tag{20}$$

Scatter matrices coupled with L are defined as follows:

$$S_w^L = \sum_{i=1}^{k} \sum_{X \in \Pi_i} (X - M_i)^T L L^T (X - M_i), \tag{21}$$

$$S_b^L = \sum_{i=1}^{k} n_i (M_i - M)^T L L^T (M_i - M). \tag{22}$$

In this way, with obtained L it can be computed the optimal R by solving an optimization problem:

$$max_R\ trace\Big(\big(R^T S_w^L R\big)^{-1}\big(R^T S_b^L R\big)\Big). \tag{23}$$

This problem can be solved as an eigenvalue problem:

$$S_w^L \mathbf{x} = \lambda S_b^L \mathbf{x}. \tag{24}$$

The optimal R can be then obtained by applying an eigendecomposition to matrix resulting from:

$$\big(S_w^L\big)^{-1} S_b^L. \tag{25}$$

It should be noted that the sizes of scatter matrices in 2DLDA are much smaller that those in LDA. Specifically, the size of S_w^R and S_b^R is $r \times r$ and the size of S_w^L and S_b^L is $c \times c$.

3.2. Pseudocode of 2DLDA algorithm

```
1. Compute the mean $M_i$ of $i$th class for each $i$ as $M_i = \frac{1}{n_i}\sum_{X\in\Pi_i} X$;
2. Compute the global mean as $M = \frac{1}{n}\sum_{i=1}^{k}\sum_{X\in\Pi_i} X$;
3. $R_0 \leftarrow$ identity matrix;
4. For $j$ from $1$ to $I$
5.
   $$S_w^R \leftarrow \sum_{i=1}^{k}\sum_{X\in\Pi_i}(X - M_i)R_{j-1}R_{j-1}^T(X - M_i)^T, \tag{26}$$
   $$S_b^R \leftarrow \sum_{i=1}^{k} n_i(M_i - M)R_{j-1}R_{j-1}^T(M_i - M)^T; \tag{27}$$
6. Compute the first $l_1$ eigenvectors $\{\phi_l^L\}_{l=1}^{l_1}$of $(S_w^R)^{-1}S_b^R$;
7. $L_j \leftarrow [\phi_1^L, \dots, \phi_{l_1}^L]$;
8.
   $$S_w^L \leftarrow \sum_{i=1}^{k}\sum_{X\in\Pi_i}(X - M_i)^T L_j L_j^T (X - M_i), \tag{28}$$
   $$S_b^L \leftarrow \sum_{i=1}^{k} n_i(M_i - M)^T L_j L_j^T (M_i - M); \tag{29}$$
9. Compute the first $l_2$ eigenvectors $\{\phi_l^R\}_{l=1}^{l_2}$of $(S_w^L)^{-1}S_b^L$;
10. End for
11. $L \leftarrow L_I, R \leftarrow R_I$;
12. $B_l \leftarrow L^T A_l R$, for $l = 1, \dots, n$;
13. return($L, R, B_1, \dots, B_n$).
```

The most time consuming steps in 2DLDA computing are lines 5, 8 and 13. The algorithm depends on the initial choice of R_0. In [19] it was showed and recommended to choose an identity matrix as R_0.

4. Principal component analysis

Principal component analysis (PCA) [9] is a linear feature transformation and dimensionality reduction method, which maps the n-dimensional input possibly correlated data to K-dimensional ($K < n$) linearly uncorrelated variables (mutually independent principal components) with respect to the variability. PCA converts the data by a linear orthogonal transformation using the first few principal components, which usually represent about 80% of the overall variance. The principal component basis minimizes the mean square error of approximating the data. This linear basis can be obtained by application of an eigendecomposition to the global covariance matrix estimated from the original data.

4.1. Mathematical description

The characteristic mathematical stages of PCA can be briefly described as follows [2, 9]. Firstly suppose that the training data are represented by M n-dimensional feature vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$. One of the integral parts of PCA is the centering of all vectors (subtracting the mean) as:

$$\mathbf{\Phi}_i = \mathbf{x}_i - \bar{\mathbf{x}}, \quad i \in \langle 1; M \rangle, \tag{30}$$

where

$$\bar{\mathbf{x}} = \frac{1}{M} \sum_{i=1}^{M} \mathbf{x}_i \tag{31}$$

is the training mean vector. From the centered vectors $\mathbf{\Phi}_i$ the centered data matrix with dimension $n \times M$ is created as:

$$A = [\mathbf{\Phi}_1 \mathbf{\Phi}_2 \ldots \mathbf{\Phi}_M]. \tag{32}$$

To represent the variance of the data across different dimensions, the global covariance matrix is computed as:

$$C = \frac{1}{M-1} \sum_{i=1}^{M} \mathbf{\Phi}_i \mathbf{\Phi}_i^T = \frac{1}{M-1} \sum_{i=1}^{M} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T = \frac{1}{M-1} A A^T. \tag{33}$$

An eigendecomposition is applied to the covariance matrix in order to obtain its eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$ and corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ and it satisfies the linear equation:

$$\mathbf{C}\mathbf{u}_i = \lambda_i \mathbf{u}_i, \quad i \in \langle 1; n \rangle. \tag{34}$$

The principal components are determined by K leading eigenvectors resulting from the decomposition. The dimensionality reduction step is performed by keeping only the eigenvectors corresponding to the K largest eigenvalues ($K < n$). These eigenvectors form the transformation matrix U_K with dimension $n \times K$:

$$U_K = [\mathbf{u}_1 \mathbf{u}_2 \ldots \mathbf{u}_K], \tag{35}$$

while $\lambda_1 > \lambda_2 > \ldots > \lambda_n$. Finally, the linear transformation $\mathbf{R}_n \to \mathbf{R}_K$ is computed according to Equation (1) as:

$$\mathbf{y}_i = U_K^T \mathbf{\Phi}_i = U_K^T (\mathbf{x}_i - \bar{\mathbf{x}}), \quad i \in \langle 1; M \rangle. \tag{36}$$

where $\mathbf{y}_i$ represents the transformed feature vector. The value of K can be chosen as needed or according to the following comparative criterion:

$$\frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{n} \lambda_i} > T, \tag{37}$$

where the threshold $T \in \langle 0.9; 0.95 \rangle$. Since

$$\sum_{i=1}^{n} \lambda_i = trace(U), \tag{38}$$

the comparative criterion can be rewritten as:

$$\frac{\sum_{i=1}^{K} \lambda_i}{trace(U)} > T. \tag{39}$$

4.2. Classical PCA in ASR

In this section we describe PCA trained from the whole amount of training data (see Section 5.1). Two kinds of input data for PCA were used. The first kind was represented by 26-dimensional LMFE features and the second one by the 13-dimensional MFCCs. Each parametrized speech signal in the corpus is represented by a LMFE or MFCC matrix $X^{(i)}$, $i \in \langle 1; N \rangle$ with dimension $26 \times n_i$ (or $13 \times n_i$, see Section 5.2), where n_i represents the number of frames in i-th recording and N represents the number of training speech signals (N=36917).

At the first stage, the initial data preparation is performed, which requires the mathematical computations described by Equations 30-32. The global covariance matrix is computed according to Equation 33 and then decomposed to a set of eigenvector-eigenvalue pairs. According to the K largest eigenvalues the corresponding eigenvectors were chosen. These ones formed the transformation matrix U_K (see Equation 35), which was used to transform the train and test corpus into PCA feature space.

Note that the final dimension (K) of the feature vectors after PCA transformation was chosen independently from the criterion formula (Equation 37). Detailed reasons are given in Sections 5.2 and 6.3. However, for interest, the determined optimal dimensions for different PCA configurations computed by Equation 37 are listed in Section 6.3.

4.3. Partial-data trained PCA

In case of relatively small training corpus there is no problem to compute the covariance matrix. But, in case of large corpora (thousands of recordings) and high-dimensional data there may occur a problem related to processing time ($\approx$ several hours) consumption and memory requirements ($\approx 20GB$). We found that for PCA learning is not necessary to use the whole training data but it may be sufficient a part of them [16]. In other words, PCA can be trained from limited (reduced) amount of training data, while the performance is maintained, or even improved. We called this procedure as *Partial-data trained PCA*.

Partial-data PCA training can be viewed as a kind of feature selection process. The main idea is to select the statistically significant data (feature vectors) from the whole amount of training data. There are two major processing stages. The first stage is the data selection based on PCA separately applied to all training feature vectors. Suitable vectors are concatenated into one train matrix, which is treated as the input for the main PCA. The second stage is the main PCA (see Section 4.1).

Suppose now that apply the same conditions as in Section 4.1. Then the selection process based on PCA (without projecting phase) can be described as follows. Each 26-dimensional LMFE (or 13-dimensional MFCC) feature vector $\mathbf{x}_i,\ i \in \langle 1; M\rangle$ (see Section 5.2) is reshaped to its matrix version $X_i,\ i \in \langle 1; M\rangle$ with dimension 2×13 (in case of MFCC vectors, the 13-dimensional vector was extended with zero coefficient in order to reshape to matrix with dimension 2×7). After mean subtraction the covariance matrix is computed as:

$$C_i = \frac{1}{k-1} X_i X_i^T, \quad i \in \langle 1; M\rangle;\ k = 13 \text{ (for MFCC, } k = 7). \tag{40}$$

In the next step, the eigendecomposition is performed on the covariance matrix C_i, which results in i sets of eigenvectors $\mathbf{w}_{i1}, \mathbf{w}_{i2}$ and eigenvalues α_{i1}, α_{i2}:

$$C_i \mathbf{w}_{ij} = \alpha_{ij} \mathbf{w}_{ij}, \quad i \in \langle 1; M\rangle,\ j \in \langle 1; 2\rangle, \tag{41}$$

where

$$W_i = [\mathbf{w}_{i1} \mathbf{w}_{i2}]. \tag{42}$$

Note that the parameters $\mathbf{w}_{i1}, \mathbf{w}_{i2}$ and α_{i1}, α_{i2} at each iteration i are updated with new parameters resulting from a new eigendecomposition. For PCA-based selection the eigenvectors $\mathbf{w}_{i1}, \mathbf{w}_{i2}$ are not used. On the other hand, the eigenvalues α_{i1}, α_{i2} are the key elements because the selective criterion is based exactly on them. Using these eigenvalues, the percentage proportion P_i is computed as:

$$P_i = \frac{\alpha_{i1}}{\sum\limits_{j=1}^{2} \alpha_{ij}} = \frac{\alpha_{i1}}{\alpha_{i1} + \alpha_{i2}} = \frac{\alpha_{i1}}{trace(C_i)}, \tag{43}$$

which determines the percentage of the variance explained by the first eigenvalue in the eigenspectrum. Further, it is necessary to choose a threshold T. It can be chosen from two different intervals. The first one is defined as $T_1 \in \langle 50; \approx 65\rangle$ and the second one as $T_2 \in \langle \approx 85; 99.9\rangle$. Then the selective criterion can be based on the following logical expressions:

$$P_i \leq T_1 \tag{44}$$

for the first interval, or

$$P_i \geq T_2 \tag{45}$$

for the second interval. If the evaluation of the expression yields a logical true then the current feature vector is classified as statistically significant for PCA training. This vector is stored and the selection continues for the next vector. In this way, the whole training corpus is processed. From the selected vectors a training matrix is composed, which is treated as the input for the main PCA described in Section 4.1. As was mentioned in Section 4.1, there are M training vectors in the corpus. If the selected subset contains M' vectors ($M' \ll M$) then the Equation 32 can be modified as:

$$A' = [\phi_1 \phi_2 \dots \phi_{M'}], \tag{46}$$

where ϕ_i is the mean subtracted feature vector in the new train matrix. The next mathematical computations are identical with Equations 33-36. The partial-data training procedure for LMFE feature vectors is illustrated in the Figure 4.3. Note that for MFCC-based partial-data PCA the figure would be analogous with the Figure 4.3.

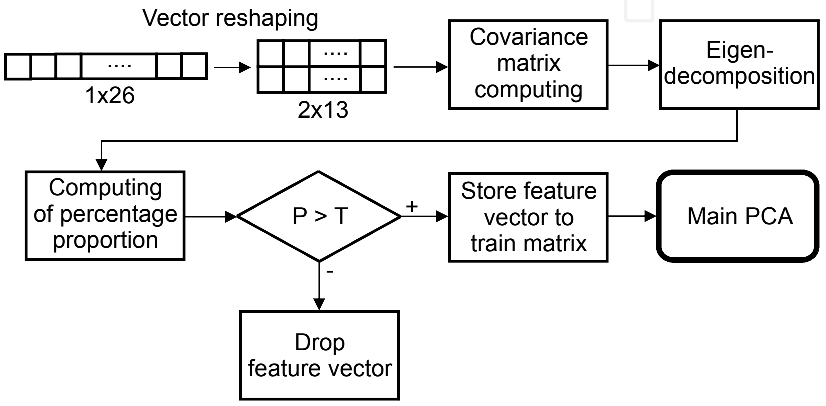

Vector reshaping
1x26
2x13
Covariance matrix computing
Eigen-decomposition
Computing of percentage proportion
P > T
+
-
Store feature vector to train matrix
Main PCA
Drop feature vector

Figure 1. Block diagram of the partial-data PCA training procedure

The new train matrix can be viewed as a radically-reduced, more relevant representation of the training corpus. It has a nearly homoscedastic variance structure because it contains only those feature vectors, which have almost the same variance distribution. Feature vectors selected from the interval represented by threshold T_1 can be characterized as data clusters, which have very small variance distribution explained by the first eigenvalue among the direction of the corresponding first eigenvector. On the other hand, the feature vectors from the interval represented by threshold T_2 are clusters, which have large variance distribution among the first eigenvector. In both cases, the largeness of the variance is determined by the first eigenvalue. The size of the selected partial data set depends on the value of T_1 or T_2. The size of partial set can be expressed in percentage amount as:

$$subset\_size = \frac{M'}{M} \times 100. \tag{47}$$

We found that a practical importance has a ratio, when

$$\frac{M'}{M} \in \langle 0.001; 0.15 \rangle, \tag{48}$$

so the selected subset contains maximally 15% of data of the whole training data amount. For example, there are approximately 19 million training vectors in our corpus. According to Equation 48 it is sufficient to extract $\approx$ 19000 vectors for partial-data training. But, as it will be showed in Section 6.3.2 this argument does not apply to all cases. The time cunsumption and memory costs of the covariance matrix computation of the reduced data set are much smaller than the costs of the covariance matrix computation in case of the whole corpus. In case of partial-data training it is needed to allocate the memory only for one investigated feature vector and for the other data elements for mathematical computations. These memory requirements are of order of units of megabytes. In other words, the advantage of the partial-data training is that it does not require the loading of the whole data matrix in the main memory.

5. Speech corpus and experimental conditions

5.1. Speech corpus

All experiments were evaluated by using a Slovak speech corpus *ParDat1* [5], which contains approx. 100 hours spontaneous parliamentary speech recorded from 120 speakers (90% of men). For acoustic modeling 36917 training utterances were exactly used. For testing purposes 884 utterances were used.

5.2. Speech preprocessing

The speech signal was preemphasized and windowed using Hamming window. The window size was set to 25ms and the step size was 10ms. Fast Fourier transform was applied to the windowed segments. Mel-filterbank analysis with 26 channels was followed by logarithm application to the linear filter outputs. This processing resulted in 26-dimensional LMFE features, which were used for PCA-based processing.

In case of MFCC baseline feature extraction, the LMFE vectors were further decorrelated by discrete cosine transform (DCT). The first 12 MFCCs were retained and augmented with the 0-th coefficient. During the acoustic modeling the first and second order derrivatives were computed and added to the basic vectors. Thus, the final MFCC vectors were 39-dimensional.

For LDA and 2DLDA-based processing the 13-dimensional MFCC vectors were used as the input for these methods. In order to regular comparison of recognition accuracy levels in the evaluation process all of LDA and 2DLDA models were trained using 39-dimensional LDA (2DLDA) vectors. In the evaluation, the 39-dimensional MFCC models were treated as reference models so the dimensions were identical. The number of classes k used in LDA and 2DLDA were identical with the number of phonetic classes in acoustic modeling ($k = 45$).

5.3. Acoustic modeling

Our recognition system used context independent monophones modeled using a three-state left-to-right HMMs. The number of Gaussian mixtures per state was a power of 2, starting from 1 to 256. The phone segmentation of 45 phones was obtained from embedded training and automatic phone alignment. The number of trained monophone models corresponded to the number of phonemes and basic classes for LDA and 2DLDA. For testing purposes a word lattice was created from a bigram language model. The language model was built from the test set. The vocabulary size was 125k. The feature extraction, HMM training and testing by using HTK (Hidden Markov Model) Toolkit [20] were performed.

5.4. Evaluation

In order to evaluate the experiments we chose the accuracy as the evaluation parameter. Accuracies were computed as the ratio of the number of all word matches (resulting from the recognizer) to the number of the reference words [20]. In all experiments the accuracy is given in percentage.

6. Experiments and results

This section is a major part of the whole chapter. It provides a detailed and extensive experimental evaluation of the performance of the mentioned linear transformation methods and their combinations. The section presents the results of the recognition accuracy levels resulting from different experimental configurations.

6.1. Conventional LDA-based processing

In this section, the conventional LDA is investigated. The LDA-based statistical computing was performed according to mathematical description of Equations 3–12 in Section2.1. Note that the class label of each supervector composed according to Equation 8 was assigned to it according to the class label of the current basic vector $\mathbf{x}[j]$ at the center position j. In our experiments we tried 5 lengths J of supervector; $J = 3, 5, 7, 9$ and 11. This means that the dimensions of the covariance matrices in the statistical estimation were 39×39, 65×65, 91×91, 117×117 and 143×143. As it was mentioned in Section 2.1, in case when the length of supervector was greater than the number of classes, the between-class scatter (covariance) matrices were close to singular. From this reason we used for these cases the computation of Σ_B according to Equation 9.

6.1.1. Supervector compositions and the scatter matrices

It is known that the covariance (scatter) matrices are in general symmetric square positive-definite regular matrices. These arguments apply also for matrices in LDA. Since in LDA the covariance matrices are computed from supervectors, there may occur a problem with the symmetry of these matrices. We found that the symmetry depends on the way, in which the supervectors are constructed. The Figure 2 illustrates two types of supervector construction with example of vector length 4. The subfigure (a) illustrates the classical way of construction of supervector by using a simple concatenation. The subfigure (b) illustrates a construction, where the final structure of the supervector is preserved according to the structure of the basic vectors. Thus, if the first few coefficients of the basic vector preserve a higher energy than the coefficients with lower order, then the new supervector follows this tendency.

It should be noted that the arrangement of the coefficients in the supervector impacts the symmetry of the matrices and this can affect other properties. These facts are proven in Figure 3. From these figures it can be seen the influence of the supervector construction to the symmetry of the scatter matrices. Figures 3 (a) and (c) represent the within-class scatter matrices in case, when the supervectors are constructed according to Figure 2 (a). It can be seen that these matrices are multisymmetric. On the other hand, the matrices in Figures 3 (b) and (d) are purely symmetric. They were computed from supervectors constructed according to Figure 2 (b).

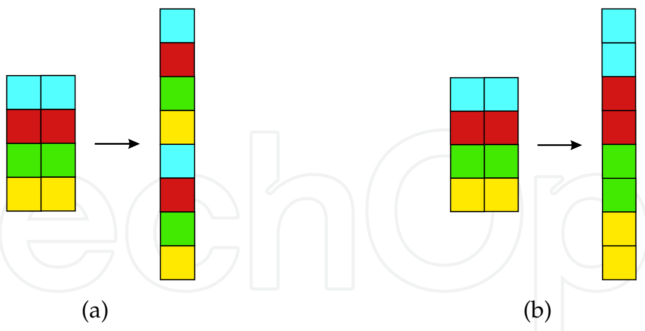


Figure 2. Different types of supervector composition; (a) composition with simple concatenating, (b) composition with retaining the structure of the basic vector

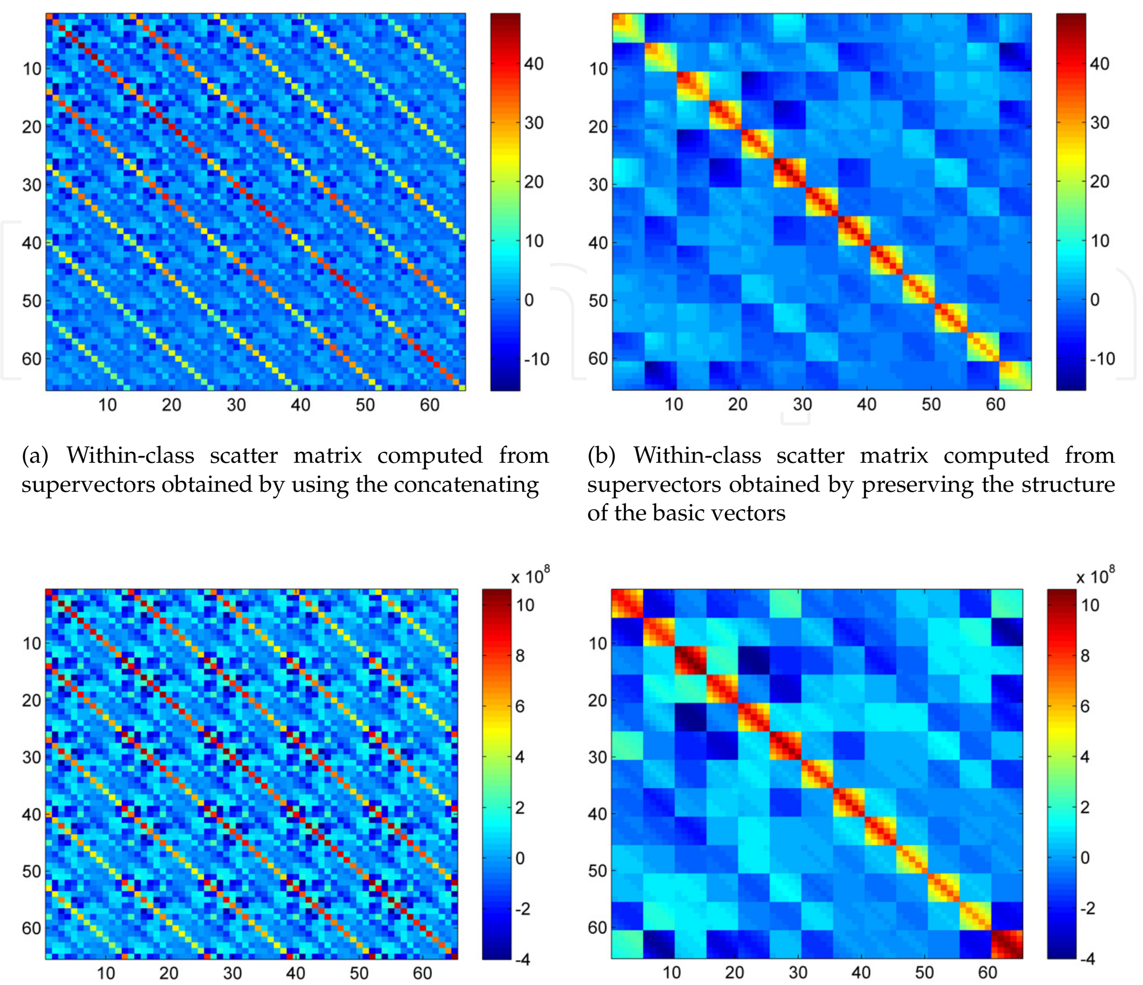
6.1.2. Between-class scatter matrix and the singularity

As was mentioned in Section 2.1, the between-class scatter matrices for context length greater than $J = 3$ were computed according to Equation 9 instead of the classical Equation 6. The Figure 4 (a) demonstrate that the between-class scatter matrix computed for context length $J = 5$ according to Equation 6 is not symmetric. In addition it is computed from supervectors constructed according to Figure 2 (a). The Figure 4 (b) illustrates a similar case as in Figure 4 (a). This matrix is computed from supervectors constructed according to Figure 2 (b). It can be seen that this matrix si only close to symmetric and in the statistical estimation this can result in singular between-class matrix and complex valued numbers in the transformation LDA matrix. Note that the symmetric between-class scatter matrices in Figure 3 were computed according to Equation 9.

6.1.3. Results

The experiments based on LDA can be divided into three categories related to dimension of the LDA transformation matrix. The first category is represented by LDA matrix with dimension 13×39. Thus, for transformation were retained only the first 13 eigenvectors corresponding to 13 leading eigenvalues. The final dimension of the features were expanded to 39 with Δ and $\Delta\Delta$ coefficients. The second category is represented by LDA matrix with dimension 19×39 so for transformations were used more LDA coefficients. Note that the final dimension of features was 38 $(19 + \Delta)$. The third category is represented by LDA matrix with dimension 39×39 and in this case were not used the Δ and $\Delta\Delta$ coefficients. The difference between these three categories is that for acoustic modeling were used various numbers of dimensions and data-dependent and data-independent Δ and $\Delta\Delta$ coefficients. The LDA coefficients with lower order (14–39) can be viewed as Δ and $\Delta\Delta$ coefficients estimated in data-dependent manner. The experimental results for LDA are given in the Table 1. The results are analyzed separately for the mentioned categories.

1. The highest accuracies were achieved for 13 LDA coefficients expanded with Δ and $\Delta\Delta$ coefficients and for $J = 3$. The maximum improvement compared to MFCC model is +2.05% for 4 mixtures. Only for 1 mixture any improvement was achieved.



(a) Within-class scatter matrix computed from supervectors obtained by using the concatenating

(b) Within-class scatter matrix computed from supervectors obtained by preserving the structure of the basic vectors

(c) Between-class scatter matrix computed from supervectors obtained by using the concatenating

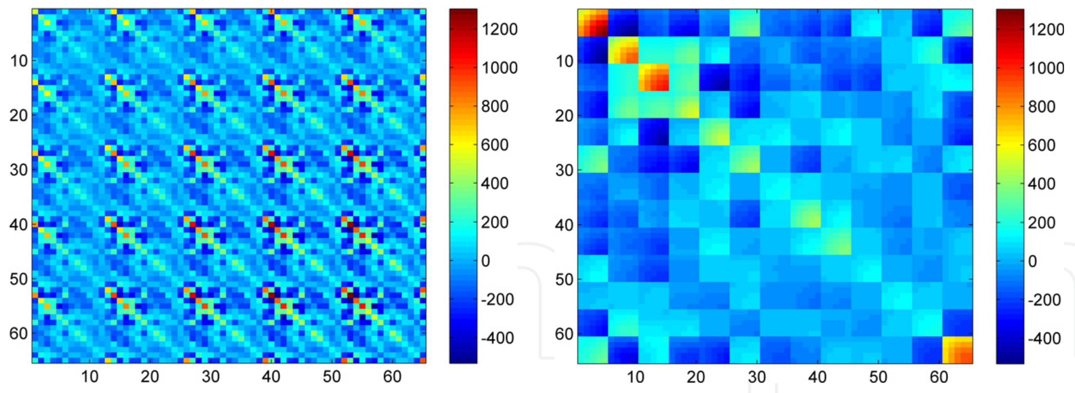
(d) Between-class scatter matrix computed from supervectors obtained by preserving the structure of the basic vectors

Figure 3. Within-class and between-class scatter matrices computed from supervectors with length 65 composed in different ways

2. In case of LDA matrix with dimension 19×39 the improvement is lower than in the previous case. The performance only for 2, 4, 8 and 256 mixtures was improved. It can be also seen that for 256 mixtures the improvement for higher context length was achieved. Note that acoustic models in this experiment have smaller dimension as the reference model ($38 < 39$).
3. The results in the last case, when the dimension of LDA matrix was 39×39 are not satisfactory. In all cases, the performance was decreased. But we can conclude that the longer lengths of context are suitable for higher dimensions of transformation matrix (without Δ and $\Delta\Delta$).

6.2. 2DLDA-based processing

In this section we extensively evaluate the performance of 2DLDA at different configurations and compare with the reference MFCC model and also with the performance of conventional



(a) Between-class scatter matrix computed from supervectors constructed according to Figure 2 (a)

(b) Between-class scatter matrix computed from supervectors constructed according to Figure 2 (b)

Figure 4. Close to symmetric between-class scatter matrices computed according to Equation 6 for context length $J = 5$

Number of mixtures	1	2	4	8	16	32	64	128	256
MFCC model (39-dim.)	82.32	83.26	85.06	87.77	89.53	90.83	91.48	92.37	92.50
13 LDA$+\Delta+\Delta\Delta$ (39-dim.)	81.37	83.60	87.11	88.47	90.03	90.88	91.80	92.48	92.90
Abs. difference	−0.95	+0.34	+2.05	+0.70	+0.50	+0.05	+0.32	+0.11	+0.40
Context length J	J=3	J=3	J=3	J=3	J=3	J=3	J=3	J=3	J=3
Supervector length	39	39	39	39	39	39	39	39	39
19 LDA $+\Delta$ (38-dim.)	82.02	83.46	85.97	88.27	89.32	90.45	91.37	82.18	92.65
Abs. difference	−0.30	+0.20	+0.91	+0.50	−0.21	−0.38	−0.11	−0.19	+0.15
Context length J	J=3	J=3	J=3	J=3	J=5	J=7	J=5	J=5	J=5
Supervector length	39	39	39	39	65	91	65	65	65
39 LDA (39-dim.)	79.82	81.41	83.31	85.13	86.83	88.19	89.10	89.98	90.69
Abs. difference	−2.50	−1.85	−1.75	−2.64	−2.70	−2.64	−2.38	−2.39	−1.81
Context length J	J=3	J=3	J=5	J=7	J=7	J=5	J=5	J=7	J=7
Supervector length	39	39	65	91	91	65	65	91	91
Max. accuracy of LDA	82.02	**83.60**	**87.11**	**88.47**	**90.03**	**90.88**	**91.80**	**92.48**	**92.90**
Max. abs. difference	−0.30	**+0.34**	**+2.05**	**+0.70**	**+0.50**	**+0.05**	**+0.32**	**+0.11**	**+0.40**

Table 1. Accuracy levels (%) for conventional LDA with different number of retained dimensions (13, 19 and 39) compared to baseline MFCC model

LDA reported in Section 6.1.3. The whole mathematical 2DLDA computing was performed according to Equations 13–25. The statistical estimations are similar as in conventional LDA. The main difference is that it is necessary to compute two eigendecompositions and we have two transformation matrices; L and R. 2DLDA does not deal with supervectors as in LDA but with supermatrices, which are the basic data elements in 2DLDA (instead of vectors). These supermatrices were created from the basic cepstral vectors by coupling them together. Similarly as in LDA, we used 5 different sizes of supermatrices according to the number of contextual vectors (context size J). Thus, the sizes of supermatrices were 13×3, 13×5, 13×7,

13×9 and 13×11. Consequently, the class mean, global mean, within-class scatter matrix and between-class scatter matrix have corresponding sizes according to the current length of context. For example, when the context size J was set to 7, in statistical estimation 7 cepstral vectors were coupled together to form a supermatrix 13×7. Then, the statistical estimators have the following dimensions:

- class means $M_i : 13 \times 7$,
- global mean $M : 13 \times 7$,
- left within-class scatter matrix $S_w^L : 7 \times 7$,
- left between-class scatter matrix $S_b^L : 7 \times 7$,
- right within-class scatter matrix $S_w^R : 13 \times 13$,
- right between-class scatter matrix $S_b^R : 13 \times 13$,
- left transformation matrix $L : 13 \times 13$,
- right transformation matrix $R : 7 \times 7$.

The mathematical computations resulted in the transformations L and R. These matrices were then used to transform the whole speech corpus. In this way, each supermatrix created from the coupled vectors in the recording was transformed to its reduced version. The dimension reduction step was done by choosing the required size of L and R. In the next step, each transformed supermatrix was re-transformed to vector according to the matrix-to-vector alignment. The specific dimensions used in transformations are listed in the Table 2. Since the mathematical part of 2DLDA is an iteration algorithm it was necessary to set the number of iterations I. In [19] it is recommended to run the iteration loop only once ($I = 1$), which significantly reduces the total running time of the algorithm. In our 2DLDA experiments we run the for loop three times ($I = 3$).

The results of 2DLDA performance can be divided into three categories, similarly as in case of LDA and are given in the Table 2.

1. The first category is represented by vector of dimension 13, which resulted from transformation. The final dimension was 39 (13 2DLDA $+\Delta + \Delta\Delta$ coefficients). As it can be seen from the Table 2, this case resulted in the highest accuracies for 2DLDA with context length $J = 3$.
2. The second category is represented by vector of dimension 19. The final dimension was 38 (19 2DLDA $+\Delta$ coefficients). Note that for example in case of transformed supermatrix with dimension 10×2 to obtain a vector with dimension 19, the last coefficient in the matrix-to-vector alignment was ignored. From the Table 2 it can be seen that 2DLDA at this dimension does not perform successfully. The performance of the base MFCC model was not improved.
3. For the third category applies similar conclusions as in the previous case. In these experiments the feature vector dimension was 39 (without Δ and $\Delta\Delta$ coefficients).

The maximum improvement achieved by 2DLDA was +2.01% for context length $J = 3$ and for one iteration ($I = 1$).

Number of mixtures	1	2	4	8	16	32	64	128	256
MFCC model (39-dim.)	82.32	83.26	85.06	87.77	89.53	90.83	91.48	92.37	92.50
13 2DLDA$+\Delta+\Delta\Delta$ (39-dim.)	82.67	84.60	87.07	88.87	90.28	91.16	91.70	92.46	92.82
Abs. difference of 2DLDA	+0.35	+1.34	+2.01	+1.10	+0.75	+0.33	+0.22	+0.09	+0.32
Abs. difference of LDA	−0.95	+0.34	+2.05	+0.70	+0.50	+0.05	+0.32	+0.11	+0.40
Context length J of 2DLDA	J=3	J=3	J=3	J=3	J=3	J=3	J=3	J=3	J=3
Supermatrix full size	13×3	13×3	13×3	13×3	13×3	13×3	13×3	13×3	13×3
Retained matrix $(L \times R)$	13×1	13×1	13×1	13×1	13×1	13×1	13×1	13×1	13×1
Num. of iterations I	I=3	I=1	I=1	I=1	I=1	I=1	I=1	I=1	I=1
19 2DLDA $+\Delta$ (38-dim.)	79.35	81.90	84.03	86.42	88.31	89.63	90.77	91.41	92.19
Abs. difference of 2DLDA	−2.97	−1.36	−1.03	−1.35	−1.22	−1.20	−0.71	−0.96	−0.31
Abs. difference of LDA	−0.30	+0.20	+0.91	+0.50	−0.21	−0.38	−0.11	−0.19	+0.15
Context length J of 2DLDA	J=5	J=5	J=5	J=5	J=5	J=5	J=5	J=5	J=5
Supermatrix full size	13×5	13×5	13×5	13×5	13×5	13×5	13×5	13×5	13×5
Retained matrix $(L \times R)$	10×2	10×2	10×2	7×3	7×3	10×2	7×3	7×3	10×2
Num. of iterations I	I=3	I=1	I=1	I=3	I=1	I=1	I=2	I=1	I=1
39 2DLDA (39-dim.)	80.13	81.51	83.62	85.78	87.62	88.91	90.15	91.00	91.66
Abs. difference of 2DLDA	−2.19	−1.75	−1.44	−1.99	−1.91	−1.92	−1.33	−1.37	−0.84
Abs. difference of LDA	−2.50	−1.85	−1.75	−2.64	−2.70	−2.64	−2.38	−2.39	−1.81
Context length J of 2DLDA	J=3	J=5	J=3	J=5	J=5	J=5	J=5	J=5	J=7
Supermatrix full size	13×3	13×5	13×3	13×5	13×5	13×5	13×5	13×5	13×7
Retained matrix $(L \times R)$	13×3	13×3	13×3	13×3	13×3	13×3	13×3	13×3	10×4
Num. of iterations I	I=1	I=3	I=1	I=3	I=3	I=1	I=1	I=1	I=1
Max. accuracy of LDA	82.02	83.60	87.11	88.47	90.03	90.88	91.80	92.48	92.90
Max. abs. difference	−0.30	+0.34	+2.05	+0.70	+0.50	+0.05	+0.32	+0.11	+0.40
Max. accuracy of 2DLDA	**82.67**	**84.60**	87.07	**88.87**	**90.28**	**91.16**	91.70	92.46	92.82
Max. abs. difference	+**0.35**	+**1.34**	+2.01	+**1.10**	+**0.75**	+**0.33**	+0.22	+0.09	+0.32

Table 2. Accuracy levels (%) for 2DLDA with different number of retained dimensions compared to baseline MFCC model and conventional LDA

6.3. PCA-based processing

In this section, we experimentally evaluate the performance of the full-data trained PCA method by using the whole amount of training data for LMFE and MFCC features. In the next part of this section we present the results of partial-data trained PCA with various parameters. Note that all of the PCA-based models were transformed with PCA matrix with dimension 13×13 and the features were then expanded with Δ and $\Delta\Delta$ coefficients. This resulted in final dimension 39.

6.3.1. Full-data trained PCA

As it was mentioned, PCA requires allocation of the whole data matrix in the memory. In addition, the covariance matrix is computed from this data matrix, which may be a computationally very difficult operation. In order to compare the partial-data trained models with the full-data trained model it was necessary to do the above mentioned computation.

The full-data trained PCA was performed on a Linux machine with 32GB memory. The training data were loaded in the memory sequentially by data blocks and then concatenated to one data matrix (see Equation 32). From this matrix the covariance matrix according to Equation 33 was computed. Then the integral parts of PCA according to Equations 34-36 were performed. In the next step, the acoustic modeling based on the PCA transformed features was done. The evaluation results of the full-data trained PCA for LMFE features are listed in the Table 5 and for MFCC features in the Table 6.

6.3.2. Partial-data trained PCA

The selective process for the feature vectors according to Fig. 1 was performed and M-times repeated. Overall, 10 partial-data trained models with LMFE features were learned. 5 models were learned for selection based on threshold T_1 and 5 ones for T_2. For MFCC features apply an identical scheme. The parameters for these models are listed in the Table 3 and Table 4. According to Equation 48, 5 subset models (0.1%, 1%, 5%, 10% and 15%) were composed.

Approx. DB size	0.1%	1%	5%	10%	15%
Num. of vectors M'	22229	187248	947804	1936764	2842838
Threshold T_1	51.40	54.05	59.10	63.00	65.75
Opt. dimension d	$d = 8$	$d = 8$	$d = 8$	$d = 8$	$d = 9$

Approx. DB size	0.1%	1%	5%	10%	15%
Num. of vectors M'	23547	206624	962434	1899584	2849321
Threshold T_2	98.00	96.10	93.20	91.00	89.20
Opt. dimension d	$d = 5$	$d = 6$	$d = 7$	$d = 8$	$d = 8$

Table 3. Parameters used for partial-data PCA models trained from LMFE

Approx. DB size	0.1%	1%	5%	10%	15%
Num. of vectors M'	21021	195034	952664	1900915	2857423
Threshold T_1	51.10	53.35	57.45	60.60	63.10
Opt. dimension d	$d = 12$	$d = 12$	$d = 12$	$d = 12$	$d = 12$

Approx. DB size	0.1%	1%	5%	10%	15%
Num. of vectors M'	20697	194742	965972	1941011	2860557
Threshold T_2	98.60	96.40	92.62	89.70	87.50
Opt. dimension d	$d = 11$	$d = 12$	$d = 12$	$d = 12$	$d = 12$

Table 4. Parameters used for partial-data PCA models trained from MFCC

One of the output parameters of the partial-data PCA is the optimal dimension d determined by Equation (37). It represents the number of principal components, which could be used to transform the input data with retaining 95% of global variance. Note that the threshold values T_1 and T_2 were determined on experimental basis. The results of the partial-data PCA models are listed in the Table 5 and Table 6 for LMFE and MFCC features, respectively. Note that the table contains only the highest accuracies chosen from all models.

From the Table 5 we can conclude that for LMFE features the selected subsets of size 0.1% and 5% are not suitable to partial-data PCA training. In addition, an improvement in comparison with full-data trained PCA was achieved only for 32–256 mixtures. The maximum absolute improvement +0.43% for 64 mixtures was achieved.

Mixtures	Acc. of full PCA	Acc. of partial-data PCA	Difference	Threshold	Part of DB
1	82.80%	82.06%	−0.74%	$T_2 = 89.2$	15%
2	84.10%	83.88%	−0.22%	$T_2 = 89.2$	15%
4	86.01%	85.93%	−0.08%	$T_1 = 63.0$	10%
8	88.88%	88.21%	−0.67%	$T_2 = 89.2$	15%
16	89.84%	89.82%	−0.02%	$T_2 = 91.0$	10%
32	90.31%	90.72%	+**0.41**%	$T_2 = 91.0$	10%
64	91.00%	91.43%	+**0.43**%	$T_2 = 91.0$	10%
128	91.72%	91.91%	+**0.19**%	$T_2 = 96.1$	1%
256	92.30%	92.60%	+**0.30**%	$T_2 = 89.2$	15%

Table 5. Accuracy levels for LMFE-based full-data and partial-data trained PCA

In case of MFCC features used as the input for partial-data PCA training, the results are more satisfactory. From the Table 6 it can be seen that for all mixtures an improvement was achieved. The maximum absolute improvement is +1.25% for 1 mixture. It could be also mentioned that for MFCC features the proposed method used the smaller selected subsets ($\approx$ 1%) in comparison with LMFE features.

Mixtures	Acc. of full PCA	Acc. of partial-data PCA	Difference	Threshold	Part of DB
1	82.35%	83.60%	+**1.25**%	$T_1 = 51.10$	0.1%
2	84.24%	84.79%	+**0.55**%	$T_1 = 53.35$	1%
4	85.94%	86.33%	+**0.39**%	$T_1 = 53.35$	1%
8	87.83%	88.08%	+**0.25**%	$T_2 = 92.62$	5%
16	89.14%	89.36%	+**0.22**%	$T_2 = 92.62$	5%
32	90.19%	90.32%	+**0.13**%	$T_2 = 89.70$	10%
64	90.90%	91.27%	+**0.37**%	$T_1 = 53.35$	1%
128	91.20%	91.78%	+**0.58**%	$T_1 = 57.45$	5%
256	91.76%	92.19%	+**0.43**%	$T_2 = 96.40$	1%

Table 6. Accuracy levels for MFCC-based full-data and partial-data trained PCA

6.4. PCA-based 2DLDA

As was mentioned in Section 1, one of the issues of this chapter is the interaction of two types of linear transformations in one experiment. More specifically, the aim of this section is to present an evaluation of the mentioned interaction of PCA and 2DLDA. In other words, in this experiment we used as the input for 2DLDA the PCA-based feature vectors instead of MFCC vectors. We wanted here to demonstrate that the PCA features have comparative properties as MFCC features and that 2DLDA trained from PCA features can achieve comparative performance as 2DLDA trained from MFCC features. The PCA training was done in two ways. The first one ist the classical full-data training and the second one is the partial-data training (see Table 7).

From the results of the experiment given in the Table 7 we can conclude the following arguments. For 4 of 9 cases the performance of 2DLDA was improved using PCA features as its input. But for 3 cases of 4 the improvement was achieved for full-data training.

Number of mixtures	1	2	4	8	16	32	64	128	256
13 2DLDA+Δ + ΔΔ (39-dim.)	82.67	84.60	87.07	88.87	90.28	91.16	91.70	92.46	92.82
13 PCA+2DLDA+Δ + ΔΔ(39-dim.)	82.26	84.50	87.17	89.03	90.44	91.10	91.95	92.43	92.69
Part of DB	5%	5%	100%	100%	100%	100%	10%	10%	1%
Type of threshold	T_1	T_2	–	–	–	–	T_1	T_1	T_1
Abs. difference	−0.41	−0.10	**+0.10**	**+0.16**	**+0.16**	−0.06	**+0.25**	−0.03	−0.13

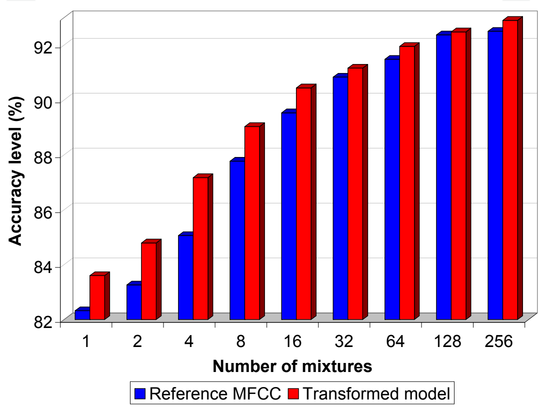
Table 7. Accuracy levels (%) of PCA-based 2DLDA

6.5. Global experimental evaluation of all methods

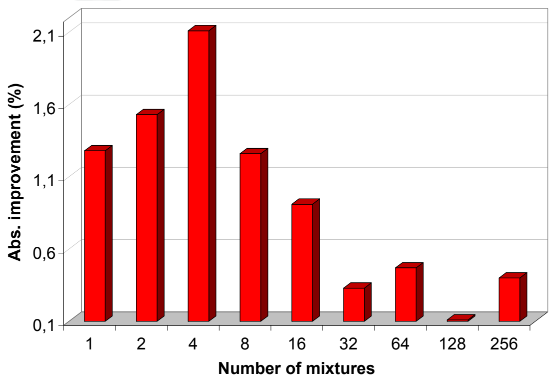
In the last section we conclude the experimental results presented in the whole chapter. Overall, we present seven types of experiments evaluating the performance of some kind of linear feature transformation applied in feature extraction in Slovak phoneme-based continuous speech recognition. Each result of the partial experiment is summarized and compared with the other results in the Table 8. The graphical comparison is given in Figure 5.

Number of mixtures	1	2	4	8	16	32	64	128	256
Conventional LDA	82.02	83.60	87.11	88.47	90.03	90.88	91.80	**92.48**	**92.90**
2DLDA	82.67	84.60	87.07	88.87	90.28	**91.16**	91.70	92.46	92.82
Full-data PCA (LMFE)	82.80	84.10	86.01	88.88	89.84	90.31	91.00	91.72	92.30
Full-data PCA (MFCC)	82.35	84.24	85.94	87.83	89.14	90.19	90.90	91.20	91.76
Partial-data PCA (LMFE)	82.06	83.88	85.93	88.21	89.82	90.72	91.43	91.91	92.60
Partial-data PCA (MFCC)	**83.60**	**84.79**	86.33	88.08	89.36	90.32	91.27	91.78	92.19
PCA+2DLDA	82.26	84.50	**87.17**	**89.03**	**90.44**	91.10	**91.95**	92.43	92.69
MFCC (reference)	82.32	83.26	85.06	87.77	89.53	90.83	91.48	92.37	92.50
Max. of transformed model	**83.60**	**84.79**	**87.17**	**89.03**	**90.44**	**91.16**	**91.95**	**92.48**	**92.90**
Abs. improvement	**+1.28**	**+1.53**	**+2.11**	**+1.26**	**+0.91**	**+0.33**	**+0.47**	**+0.11**	**+0.40**

Table 8. Global comparison of partial experiments for all types of linear transformations



(a) Comparison of transformed and MFCC models



(b) Absolute improvement of transformed models

Figure 5. Graphical global evaluation of all experiments compared to reference MFCC model

7. Conclusions and discussions

The global conclusion of the experimental part of this chapter can be divided into few following deductions.

- Principal Component Analysis can improve the performance of the MFCC-based acoustic model. As the input for PCA can be used LMFE or MFCC features.
- The proposed partial-data trained PCA achieves better results compared to full-data trained PCA. Higher improvements can be achieved in case of MFCC features used as input for partial-data PCA.
- The conventional Linear Discriminant Analysis leads to improvements almost for all mixtures, but there may occur a problem related to singularity of between-class scatter matrix in case of larger lengths of context J.
- 2DLDA achieves comparable improvements as LDA (a little bit smaller). On the other hand, it is much more stable than LDA and there is no problem with the singularity, because 2DLDA overcomes it implicitly (much smaller dimensions of scatter matrices).
- In the last step, we clearly demonstrated that the combination of PCA and 2DLDA (subspace learning) leads to further refinement and improvement compared to performance of 2DLDA.

8. Future research intentions

Based on the presented knowledge and our research intentions in the near future we would like to develop an algorithm to elimination of using the class label information (class definition) in the LDA-based experiments. In other words, we want to train the LDA and its similar supervised modifications in unsupervised way without using the labeling of speech corpus.

Acknowledgments

The research presented in this paper was supported by the Ministry of Education under the research projects VEGA 1/0386/12 and MŠ SR 3928/2010–11 and by the Slovak Research and Development Agency under the research project APVV–0369–07.

Author details

Jozef Juhár and Peter Viszlay
Technical University of Košice, Slovakia

9. References

[1] Abbasian, H., Nasersharif, B., Akbari, A., Rahmani, M. & Moin, M. S. [2008]. Optimized linear discriminant analysis for extracting robust speech features, *Proc. of the 3rd Intl. Symposium on Communications, Control and Signal Processing*, St. Julians, pp. 819–824.

[2] Bebis, G. [2003]. *Principal Components Analysis*, Department of Computer Science, University of Nevada, Reno.

[3] Belhumeur, P., Hespanha, J. & Kriegman, D. [1997]. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection, *IEEE Pattern Analysis and Machine Intelligence* 19: 711–720.

[4] Beulen, K., Welling, L. & Ney, H. [1995]. Experiments with linear feature extraction in speech recognition, *Proc. of European Conf. on Speech Communication and Technology*, pp. 1415–1418.

[5] Darjaa, S., Cerňak, M., Š. Beňuš, Rusko, M., Sabo, R. & Trnka, M. [2011]. *Rule-based triphone mapping for acoustic modeling in automatic speech recognition*, Vol. 6836 LNAI of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

[6] Duchateau, J., Demuynck, K., Compernolle, D. V. & Wambacq, P. [2001]. Class definition in discriminant feature analysis, *Proc. of European Conf. on Speech Communication and Technology, EUROSPEECH'01*, Aalborg, Denmark, pp. 1621–1624.

[7] Geirhofer, S. [2004]. Feature reduction with linear discriminant analysis and its performance on phoneme recognition, *Technical Report ECE272*, Dept. of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign.

[8] Haeb-Umbach, R. & Ney, H. [1992]. Linear discriminant analysis for improved large vocabulary continuous speech recognition, *Proc. of the IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing, ICASSP'92*, San Francisco, CA, pp. 13–16.

[9] Jolliffe, I. T. [1986]. *Principal Component Analysis*, Springer-Verlag, New York, USA.

[10] Krzanowski, W. J., Jonathan, P., Mccarthy, W. V. & Thomas, M. R. [1995]. Discriminant analysis with singular covariance matrices: methods and applications to spectroscopic data, *Applied Statistics* 44: 101–115.

[11] Kumar, N. [1997]. *Investigation of Silicon Auditory Models and Generalization of Linear Discriminant Analysis for Improved Speech Recognition*, PhD thesis, Johns Hopkins Universtiy, Baltimore, Maryland.

[12] Li, X. B. & O'Shaughnessy, D. [2007]. Clustering-based Two-Dimensional Linear Discriminant Analysis for Speech Recognition, *Proc. of the 8th Annual Conference of the International Speech Communication Association*, pp. 1126–1129.

[13] Pylkkönen, J. [2006]. LDA based feature estimation methods for LVCSR, *Proc. of the 9th Intl. Conf. on Spoken Language Processing, INTERSPEECH'06*, Pittsburgh, PA, USA, pp. 389–392.

[14] Schafföner, M., Katz, M., Krüger, S. E. & Wendemuth, A. [2003]. Improved robustness of automatic speech recognition using a new class definition in linear discriminant analysis, *Proc. of the 8th European Conf. on Speech Communication and Technology, EUROSPEECH'03*, Geneva, Switzerland, pp. 2841–2844.

[15] Song, H. J. & Kim, H. S. [2002]. Improving phone-level discrimination in LDA with subphone-level classes, *Proc. of the 7th Intl. Conf. on Spoken Language Processing, ICSLP'02*, Denver, Colorado, USA, pp. 2625–2628.

[16] Viszlay, P. & Juhár, J. [2011]. Feature selection for partial training of transformation matrix in PCA, *Proc. of the 13th Intl. Conf. on Research in Telecommunication Technologies, RTT'11*, Techov, Brno, Czech Republic, pp. 233–236.

[17] Viszlay, P., Juhár, J. & Pleva, M. [2012]. Alternative phonetic class definition in linear discriminant analysis of speech, *Proc. of the 19th International Conference on Systems, Signals and Image Processing, IWSSIP'12*, Vienna, Austria. Accepted, to be published.

[18] Yang, J., Zhang, D., Frangi, A. F. & Yang, J.-Y. [2004]. Two–Dimensional PCA: A New Approach to Appearance–Based Face Representation and Recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26: 131–137.

[19] Ye, J., Janardan, R. & Li, Q. [2005]. Two-dimensional linear discriminant analysis, *L. K. Saul, Y. Weiss and L. Bottou (Eds.): Advances in Neural Information Processing Systems* 17: 1569–1576.

[20] Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X. A., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V. & Woodland, P. [2006]. *The HTK Book (for HTK Version 3.4)*. First Published Dec. 1995.