Principal component analysis (PCA) is commonly used in feature extraction. It projects the features in direction of maximum variance. This projection can be performed in a class-dependent or class-independent manner. In this paper, we propose to optimize class-dependent PCA transformation matrix for robust MFCC feature extraction using genetic programming. For this purpose, we first map logarithm of clean speech Mel filter bank energies (LMFE) in directions of maximum variability. We obtain the mapping functions using genetic programming. After this, we form class-dependent PCA transformation matrix based on mapped LMFE and use this matrix in place of DCT in MFCC feature extraction. The experimental results show that proposed method achieves to significant isolated word recognition rate on Aurora2 database.

Index Terms: Genetic programming, PCA, Class-dependent PCA, MFCC, Speech recognition

1. Introduction

Feature extraction is a crucial step of speech recognition process which greatly affects the performance of speech recognition systems. Speech features must represent the temporal evolution of the speech spectral envelope. Some examples of common speech features are: linear prediction coefficients (LPC), perceptual linear prediction coefficients (PLP) and Mel frequency cepstral coefficients (MFCC). Among these features, MFCC feature are more commonly used for speech recognition. The MFCC features are obtained by applying discrete cosine transform (DCT) to logarithm of Mel filter bank energies (LMFE). There are several techniques that improve MFCC features from different points of view. Some approaches attempt to make MFCC more robust to channel and additive noises using weighting or compression of Mel sub-band energies [1][2]. In other group of methods, we try to overcome to disadvantages of DCT in clean or noisy conditions [3][4]. DCT is a non-adaptive procedure that projects LMFE in the direction of global variance which achieves only partial decorrelation of features. In order to overcome the partial decorrelation, several methods have been proposed to replace DCT and decorrelate LMFE. Some examples of such methods are: Principle Component Analysis (PCA) [4][5] and Independent Component Analysis (ICA) [4]. PCA, based on the principle of minimum reconstruction error, projects the data (LMFE) in the direction of maximum variability [4][5]. But, there is no guarantee that variability explained by PCA is useful for speech recognition. While PCA removes the second order dependencies of the feature vector components, ICA removes also higher order dependencies and minimizes the mutual information between the feature vector components[4]. One limitation of PCA is that it does not model nonlinear relationships among feature vector components efficiently. Several generalizations of PCA have been proposed to address this limitation. Two examples of such approaches are: nonlinear PCA (NLPCA) [5][6] and kernel PCA (KPCA) [7][8][9]. NLPCA generalizes the principal components from straight lines to curves. This can be done using neural networks with an auto-associative architecture [5]. In KPCA, a nonlinear map is used to translate nonlinear structure of features into linear ones in a feature space with a higher dimension. After this, linear PCA is applied to mapped features [7][8][9]. While NLPCA and KPCA optimize PCA transformation to overcome non-linearity in feature space, we propose to optimize class dependent PCA [13] (CD-PCA) transformation using Genetic Programming (GP) in order to project feature vector component in a space which they have maximum independence. We name this method as GCD-PCA. The feature vector components in our work are LMFE. We apply optimized CD-PCA transformation to LMFE in place of DCT. Using this transformation, LMFE features will be more uncorrelated and so their covariance matrix will be more diagonal. This results in better HMM training in each class. In addition, due to class-dependent transformation and better HMM training, we expect that these transformed LMFE are made more robust to noise. The rest of the paper is organized as follows. In section 2, we describe MFCC extraction and speech recognition based on class dependent PCA. Section 3, explains the CD-PCA optimization approach for extracting MFCC. Section 4, discusses the genetic programming and its parameters in PCA optimization method. Section 5 includes our experiments and results. Finally, we give our conclusion in section 6.

2. MFCC Extraction using Class-Dependent PCA

In conventional MFCC extraction, given a power spectrum $|X(k)|^2$ for a speech frame, the logarithm of filter bank energy $LE_j$, is calculated as[1]:

$$LE_j = \log \left( \sum_{k=1}^{L} |X(k)|^2 \phi(k) \right), \quad 1 \leq j \leq L$$

(1)

where $\phi(k)$ is $j$-th Mel triangular band-pass filter. $L$ is number of Mel filters and $k$ is a frequency index. After this, DCT is applied to log of Mel filter bank energies. DCT has the property to project the LMFE on directions of maximum global variance and so produce partially uncorrelated results. DCT is sub-optimal in data compression and decorrelation.
This is its major drawback. Due to the drawback of DCT, the other feature projection methods such as PCA and KPCA can be used in MFCC feature extraction to extract more uncorrelated LMFE. So, we first discuss the general process of PCA.

In this paper, we use CD-PCA in which PCA transformation matrix is separately determined for each class. In CD-PCA, we obtain a transformation matrix for each class, while in class independent PCA (CI-PCA), we have a single transformation matrix for all classes and data.

For obtaining CD-PCA transformation matrix, assume that we have K speech signals, \(X = \{x_1, x_2, \ldots, x_K\}\) in class i of our training set. Now suppose that, \(x_k(m)\), represents m-th frame of k-th speech signal which is pre-emphasized by hamming window, where \(1 \leq m \leq M_k\) and \(M_k\) is the total number of frames in signal \(x_k\). If number of Mel filters is equal to L as in (2), we will have a LMFE matrix of size \(L \times M_k\) for each speech signal named \(F_k\). By concatenating the matrices of speech signals, we can represent training set by matrix \(F\) of size \(L \times M\), where:

\[
M = \sum_{i=1}^{K} M_k \quad (2)
\]

Covariance matrix \(C\) of LMFE describes the correlation between LMFE. In case of CD-PCA this covariance matrix show the correlation between LMFE in a class. We use PCA to obtain a transformation matrix for applying to LMFE. As a result, the covariance matrix of transformed LMFE will be more diagonal. PCA forms a transformation matrix using eigenvalue decomposition of covariance matrix as follows. At the first, the eigenvectors are obtained from the following equation:

\[
C v_p = \lambda_p v_p \quad 1 \leq p \leq L \quad (3)
\]

where \(v_p\) is eigenvector and \(\lambda_p\) is corresponding eigenvalue. Then, PCA transformation matrix, named \(T\), is formed as following:

\[
T = [v_1, v_2, \ldots, v_L]^T, \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_L \quad (4)
\]

We apply the matrix \(T\) in place of DCT matrix to LMFE computed in equation (1), for extracting MFCC features. In this way, we obtain PCA based MFCC features.

In this paper, we use class-dependent PCA [13] in which PCA transformation matrix is separately determined for each class [13]. In class-dependent PCA, we have one covariance matrix for each class. Therefore, in speech recognition step, we use a separate transformation matrix for corresponding class which HMM represents. So, for an unknown speech sample, we extract its LMFE matrix. After that, when we want to give the matrix as observation sequence to HMM representing class i, we apply the CD-PCA transformation matrix of class i to the LMFE matrix. After this transformation, we gave transformed matrix as observation sequence to HMM of class i. The procedure is repeated for all classes. This procedure has no complexity more than conventional recognition (test) step in using HMM and Viterbi algorithm. The additional complexity is only where we multiply extracted LMFEs by \(c\) different transformation matrices corresponding to different classes. Fig. 1 shows the overall proposed speech recognition steps using class-dependent PCA.

### 3. PCA Optimization Using Genetic Programming

As said before, there is no guarantee that variability explained by PCA is useful for speech recognition. In this section, we propose to optimize PCA transformation matrix in order to obtain more independence and variability in LMFE and so speech features. For this purpose, we use genetic programming (GP) to maximize variance of LMFE and achieve to a more diagonal covariance matrix.

In order to optimize PCA transformation matrix, we first map each of LMFE using different mapping functions in directions that we obtain maximum variability. If we suppose that \(F\) represents the training LMFE feature matrix of size \(L \times M\) and \(f_i\) indicates \((i, q)\)-th entry of \(F\), we have:

\[
f_{i,q} = \theta_i(f_{i,q}) \quad 1 \leq i \leq L, \quad 1 \leq q \leq M \quad (5)
\]

where \(\theta_i\) is mapping function obtained using GP for i-th LMFE and \(f_{i,q}\) is \((i, q)\)-th entry of mapped LMFE matrix named \(\tilde{F}\). GP find mapping functions \([\theta_0, \theta_2, \ldots, \theta_L]\) such that maximize LMFE variance in a class. So, the covariance matrix of LMFE in a class becomes more diagonal. If \(\tilde{C}\) shows the covariance matrix of \(\tilde{F}\) and \(\tilde{\lambda}_p, \tilde{\lambda}_q\), represent eigenvalues and eigenvectors of matrix \(\tilde{C}\) respectively, new PCA transformation matrix is calculated as:

\[
\tilde{T} = [\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_L]^T, \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \ldots \geq \tilde{\lambda}_L \quad (7)
\]

In the next step, we project original LMFE using transformation matrix \(\tilde{T}\). As described above, we determine transformation matrix \(\tilde{T}\) in a class-dependent way similar to CD-PCA. In the recognition step, we use \(\tilde{T}\) corresponding to each class as shown in the Fig. 1(b). It can be noted that Figs 2(a), 2(b) and 2(c) show the covariance matrix of 5-th LMFE and 11-th LMFE, after applying DCT, CD-PCA and GCD-PCA, respectively. As it can be seen from these figures, the covariance matrix of these two speech features after applying GCD-PCA is more diagonal than two other methods.
4. Genetic Programming Parameters

GP is the youngest member of Evolutionary algorithms family developed by Koza [10]. Similar to genetic algorithms (GA), GP works on population of individuals and tries to find an optimal solution using genetic operators such as crossover and mutation.

GP uses tree structure chromosomes which have variable lengths. Trees can be easily evaluated in a recursive manner. Every tree node has an operator function and every terminal node has an operand, making mathematical expressions easy to evolve and evaluate. This chromosome structure in GP needs new suitable genetic operators. Crossover is applied on an individual by simply switching one of its nodes with another node from another individual in the population. With a tree-based representation, replacing a node means replacing the whole branch. This adds greater effectiveness to the crossover operator. The expressions resulting from crossover are very much different from their initial parents. Mutation affects an individual in the population. It can replace a whole node in the selected individual, or it can replace just the node’s information [10].

In our approach, we want to generate simultaneously different mapping functions using GP. Therefore, some modification in initial population and genetic operators are necessary. In the following, we briefly explain main parts of GP.

4.1. Initial population

Before first generation, a random population of 50-100 individuals should be generated. Each individual represents one solution. In this work, we want to generate \( L \) functions simultaneously. So, each individual should contain \( L \) different mapping functions where \( L \) is the number of Mel filter banks. Each function in each individual is represented by a random created tree obtained using combination of functions (called function set) and terminals (called terminal set). In this paper, we use each feature and a constant number between 0,1 as function set) and terminals (called terminal set). In this paper, we want to generate \( L \) functions simultaneously. Therefore, we should propose a new method. In this method, \( n \) random replacing of a single function with a random other function of first parent, with another random sub-tree from \( j \)-th parent, where, \( i \) and \( j \) are random numbers between 1 and \( L \).

4.2 Fitness Calculation

After generating initial population, fitness of each individual is computed using a fitness function. Fitness is a measure which represents how well an individual is. For selecting appropriate fitness functions, we should consider our algorithm objective. In this work, our main objective is to generate transformation matrix \( \hat{T} \), such that maximize variance between LMFE features in a specified class. So, for two rows of LMFE matrix \( F \), named here as \( f_p \) and \( f_q \), we can calculate our fitness function as:

\[
V = [f_p, f_q] V \quad p, q = 1, 2, ..., L
\]

\[
Fitness = \frac{2}{L(L-1)} \sum_{p=1}^{L} \sum_{q=p+1}^{L} \left( \frac{f_{pq} - f_{qp}}{2} \right)^2
\]

where \( L \) is number of Mel filters and \( M \) is as in equation (2).

4.3 Selection operator

Members of population are chosen for reproduction based on their fitness value. In this paper, we use tournament selection [11] with tournament size of 5 to select best members of population based on their fitness.

4.4 Crossover operator

Crossover operation in Genetic Programming takes two members of population and generates two new offspring. As mentioned in section 4.1, individuals in our method contain \( L \) different mapping functions. Therefore, we should propose a new crossover operator for our method. For this purpose, we propose a two-step cross-over operation. In the first step, cross-over operator interchanges a random sub-tree from \( i \)-th function of first parent, with another random sub-tree from \( j \)-th parent, where, \( i \) and \( j \) are random numbers between 1 and \( L \). As the second step, operator interchanges \( i \)-th function between two parents completely where \( i \) is a random number between 1 and \( L \).

4.5 Mutation

Mutation operation generally takes an individual and creates new offspring by altering selected individual. In this paper, individuals are function sets. So, mutation operation consists of random replacing of a single function with a random chosen member of population. Further detail can be found in [10], [11].

4.6 The new population Construction

After using genetic operators, we should decide which members of old population should be replaced by new individuals in order to generate new population. For this purpose, we use elitism method. In this method, a part of old population (Generation Gap) is copied unchanged to the next population [11]. In this paper, 10% of fittest old population are copied unchanged to the next population and remaining members of old population are replaced by new individuals. This procedure repeats until we obtain the maximum number of generations.

5. Experiments and Results

We report our results on Aurora 2 database [12] for isolated word recognition. Only clean data are used for HMM training. Our recognizer is CDHMM with 16 states and 3 Gaussian mixtures per state as [12]. There are 8 types of noises in the 3 test sets: sets A, B and C [12]. Our feature vector in all cases contains 12 projected LMFE (MFCC in case of DCT) and 12 delta-coefficients and so its length is 24.

In GCD-PCA and CD-PCA, we use only clean training set
and its covariance matrix in each class. In this way, we determine our transformation matrix in a class based on PCA and GCD-PCA.

Fig. 3 shows average word errors rate (AWER) over SNR values of 20, 15, 10, 5, 0 dB which are separated for different types of noise and three test sets. Words DCT, CD-PCA and GCD-PCA show the type of transformation matrix applied to LMFE. So, DCT indicates the conventional MFCC features. CD-PCA and GCD-PCA show cases that DCT matrix is replaced by CD-PCA and GP based optimized CD-PCA transformation matrix in MFCC features extraction. As can be seen from the figure, for all 3 test sets, GCD-PCA has lowest AWER. This is noticeable because we don’t use any information of noise in determining GCD-PCA and also CD-PCA transformation matrices. This can be due to class-dependent transformation and also more diagonal covariance matrix which cause better training of HMM in clean conditions. Additionally, in case of all three sets, CD-PCA has higher recognition rate than DCT. This is because of using class-dependent PCA transformation.

Fig. 4 shows AWER over all noise types and test sets which are separated for different SNR values. As shown in the figure, GCD-PCA decreases AWER very noticeably in comparison to CD-PCA and DCT, especially in SNR values of 0 and -5 dB. Furthermore, CD-PCA performs better than DCT in all SNR values. These results show importance and role of a class-dependent transformation such as CD-PCA and its optimization by GP.

6. Conclusion
We proposed a method for class-dependent PCA transformation matrix optimization in order to overcome drawbacks of DCT and PCA transformation. For this optimization, we used genetic programming. Using genetic programming, we determine the mapping functions that map the logarithm of Mel filter bank energies in direction of maximum variability. After mapping using these functions, we shape class-dependent PCA transformation matrix based on mapped log of Mel filter bank energies. Finally, we use this matrix in place of DCT matrix for MFCC feature extraction. The proposed optimization method performs better than DCT and class-dependent PCA for isolated word recognition rate on Aurora 2 database. Results show that using optimized class-dependent PCA matrix in MFCC extraction, significantly improve isolated word recognition rate on Aurora 2 such that average word recognition rate is about 55% higher than conventional MFCC in SNR value of 0 dB.

7. References

Figure 3. Average word error rate over SNR values (20,15,10,5,0 dB) separated for different noise types and three test sets

Figure 4. Average word error rate over all noise type and test sets separated for different SNR values