KERNEL MULTIMODAL DISCRIMINANT ANALYSIS FOR SPEAKER VERIFICATION

Min-Seok Kim, IL-Ho Yang and Ha-Jin Yu[*]

School of Computer Science, University of Seoul, Korea
{ms|heisco|hjyu}@uos.ac.kr

ABSTRACT

In this paper, we propose a robust speaker feature extraction method using kernel multimodal Fisher discriminant analysis (kernel MFDA). Kernel MFDA has been designed to have the characteristics both of kernel principal component analysis (kernel PCA) and kernel Fisher discriminant analysis (kernel FDA). Therefore, the feature vectors extracted by kernel MFDA are denoised as well as discriminated. For evaluation, we compare our proposed method with principal component analysis (PCA) and kernel PCA on the speaker verification systems.

Index Terms— Feature extraction, Speaker recognition, Speech enhancement

1. INTRODUCTION

The noise in the signal is a cause to degrade the performance of pattern recognition systems. There have been many researches on robust feature extraction such as using kernel principal component analysis (kernel PCA) [1][2][4][6]. In these researches, Kernel PCA shows good performance in noisy environments, because the feature vectors extracted by kernel PCA retain the good information for classification, while the noise components are dropped. But kernel PCA, like the other kernel method, is infeasible when the number of feature vectors increases. Speaker recognition should deal with a huge number of feature vectors which is proportional to the duration of the utterances. Therefore, we need to devise a way of using kernel PCA for the large scale problems. There are several methods to compute kernel PCA for large scale problems [9]. Among those, greedy kernel PCA [5][8] have been applied to speaker recognition, and it showed good performance [13]. In greedy kernel PCA, the computational complexity decreases by greedy filtering [5][8].

Kernel multimodal Fisher discriminant analysis, our proposed method, needs much lesser memory space than those for speaker recognition. Also our proposed method has been designed to have both of the two characteristics of kernel PCA and kernel FDA (Fisher discriminant analysis) [7]. This paper is organized as follows. In the next section, we introduce kernel PCA and greedy filtering. In section 3, we describe our proposed method. In section 4, we show the experimental results. Finally, section 5 presents the conclusions and discussions.

This work was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD)" (KRF-2007-321-A00155)

* Corresponding Author

2. FEATURE TRANSFORMATION

2.1. Kernel principal component analysis

Kernel principal component analysis (kernel PCA) [4][6] is the nonlinear version of principal component analysis (PCA). In kernel PCA, the training set $\mathbf{X} = [\mathbf{x}_1,\ldots,\mathbf{x}_l]$ in the *input space* ($\mathbf{x}_i \in \Re^d$) is mapped to an arbitrary high-dimensional *feature space* ($\phi(\mathbf{x}_i) \in F$) by the kernel function $\phi(\cdot) : \Re^d \rightarrow F$, and the normalized set (its mean is **0**) of the vectors $\mathbf{X}^F = [\phi(\mathbf{x}_1),\ldots,\phi(\mathbf{x}_l)]$ is computed as

$$\overline{\mathbf{X}}^F = \mathbf{X}^F - \frac{1}{l}\mathbf{X}^F \mathbf{1}_{l\times l} \tag{1}$$

where $\mathbf{1}_{l\times l}$ is an $l \times l$ matrix whose elements are 1's. Then the covariance matrix, $\mathbf{C}^F$ of $\overline{\mathbf{X}}^F$ is expressed as

$$\mathbf{C}^F = \frac{1}{l}\overline{\mathbf{X}}^F\,\overline{\mathbf{X}}^{F\,\prime}. \tag{2}$$

The transformation matrix of kernel PCA is derived in the same manner as PCA:

$$\lambda_i \mathbf{v}_i = \mathbf{C}^F \mathbf{v}_i . \tag{3}$$

But the eigenvectors of $\mathbf{C}^F$ cannot be computed directly because of the high dimension. The following condition is given that the eigenvectors $\mathbf{v}_1,\ldots,\mathbf{v}_{\tilde{d}}$ of $\mathbf{C}^F$ are linear combinations of $\overline{\mathbf{X}}^F$:

$$\mathbf{v}_i = \overline{\mathbf{X}}^F \boldsymbol{\alpha}_i \;\text{ for } i=1,\ldots \tilde{d} . \tag{4}$$

By multiplying $\overline{\mathbf{X}}^{F\,\prime}$ to (3) and plugging-in (2) and (4), the following equation is derived:

$$\eta_i \boldsymbol{\alpha}_i = \overline{\mathbf{K}}_{\mathbf{X}} \boldsymbol{\alpha}_i \,,\; \eta_i \equiv l\lambda_i \tag{5}$$

where $\overline{\mathbf{K}}_{\mathbf{X}}$ represents $\overline{\mathbf{X}}^{F\,\prime}\overline{\mathbf{X}}^F$. $\overline{\mathbf{K}}_{\mathbf{X}}$ is computed as

$$\overline{\mathbf{K}}_{\mathbf{X}} = \mathbf{K}_{\mathbf{X}} - \frac{1}{l}\mathbf{K}_{\mathbf{X}}\mathbf{1}_{l\times l} - \frac{1}{l}\mathbf{1}_{l\times l}\mathbf{K}_{\mathbf{X}} + \frac{1}{l^2}\mathbf{1}_{l\times l}\mathbf{K}_{\mathbf{X}}\mathbf{1}_{l\times l} \tag{6}$$

where $\mathbf{K}_\mathbf{X}$ is the kernel matrix of the training set $\mathbf{X}$. $\mathbf{K}_\mathbf{X}$ is expressed as

$$\mathbf{K}_\mathbf{X} = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & \cdots & k(\mathbf{x}_1,\mathbf{x}_l) \\ \vdots & \ddots & \\ k(\mathbf{x}_l,\mathbf{x}_1) & & k(\mathbf{x}_l,\mathbf{x}_l) \end{bmatrix}, \ \mathbf{K}_\mathbf{X} \in \Re^{l\times l}. \quad (7)$$

Here, $k(\mathbf{x}_i,\mathbf{x}_j)$ is a kernel function [4]. The coefficient $\boldsymbol{\alpha}_i$ in (4) can be computed by the eigenvectors of $\overline{\mathbf{K}}_\mathbf{X}$. Additionally, $\boldsymbol{\alpha}_i$ should be normalized to satisfy the condition [4][6] that

$$(\mathbf{v}_i\cdot\mathbf{v}_i) = \eta_i(\boldsymbol{\alpha}_i\cdot\boldsymbol{\alpha}_i) = 1, \\ \therefore \hat{\boldsymbol{\alpha}}_i = \boldsymbol{\alpha}_i / \sqrt{\eta_i} \quad (8)$$

where $\hat{\boldsymbol{\alpha}}_i$ is the normalized $\boldsymbol{\alpha}_i$. In this step, $\tilde{d}$ eigenvectors corresponding to the $\tilde{d}$ largest eigenvalues are selected:

$$\mathbf{A} = [\hat{\boldsymbol{\alpha}}_1,\ldots,\hat{\boldsymbol{\alpha}}_{\tilde{d}}]. \quad (9)$$

Finally, given test data $\mathbf{Y}=[\mathbf{y}_1,\ldots,\mathbf{y}_t]$, $\mathbf{Y}\in\Re^{d\times t}$ the transformed data onto the kernel PCA directions $\mathbf{Y}_{output}$ is computed as

$$\overline{\mathbf{K}}_\mathbf{YX} = \mathbf{K}_\mathbf{YX} - \frac{1}{l}\mathbf{K}_\mathbf{YX}\mathbf{1}_{l\times l} - \frac{1}{l}\mathbf{1}_{t\times l}\mathbf{K}_\mathbf{X} + \frac{1}{l^2}\mathbf{1}_{l\times l}\mathbf{K}_\mathbf{X}\mathbf{1}_{l\times l}, \\ \mathbf{Y}_{output} = \mathbf{A}'\overline{\mathbf{K}}_\mathbf{YX} \quad (10)$$

where $\mathbf{K}_\mathbf{YX}$ is defined to be the kernel matrix of $\mathbf{Y}$ and $\mathbf{X}$:

$$\mathbf{K}_\mathbf{YX} = \begin{bmatrix} k(\mathbf{y}_1,\mathbf{x}_1) & \cdots & k(\mathbf{y}_1,\mathbf{x}_l) \\ \vdots & \ddots & \\ k(\mathbf{y}_t,\mathbf{x}_1) & & k(\mathbf{y}_t,\mathbf{x}_l) \end{bmatrix}, \ \mathbf{K}_\mathbf{YX} \in \Re^{t\times l}. \quad (11)$$

2.2. Greedy filtering

In kernel PCA, $O(l\times l)$ memory is required to store the kernel matrix $\mathbf{K}_\mathbf{X}$, and the computational complexity increases quadratically with l. Therefore, we cannot apply it to large-scale problems such as speaker recognition. The greedy filtering [5] is one of the solutions for the large-scale problems.

If we reduce the number of training data to m in (4) to represent $\mathbf{v}_i$, we only need $O(l\times m)$ memory to compute the kernel matrix and the time for computing eigenvectors of the $m\times m$ matrix. Let $\mathbf{R}$ denotes the reduced set of $\mathbf{X}$. We can select $\mathbf{R}$ by greedy filtering: (see [5] or [8])

$$\mathbf{R} = [\mathbf{r}_1,\ldots,\mathbf{r}_m], \ \mathbf{R}\subset\mathbf{X}, \\ \mathbf{R}^F = [\phi(\mathbf{r}_1),\ldots,\phi(\mathbf{r}_m)]. \quad (12)$$

By using $\mathbf{R}^F$ instead of $\overline{\mathbf{X}}^F$ in (4), we only need the time to compute the eigenvectors of an $m\times m$ matrix.

3. THE PROPOSED METHOD

Our proposed method is the kernelized version of multimodal Fisher discriminant analysis (MFDA)[3]. MFDA is a modified Fisher discriminant analysis (FDA) which maximizes discriminant information among sub components in each class. In MFDA, new scatters are used instead of conventional between-class scatter and within-class scatter [3]. However, it is difficult to derive the kernel MFDA directly. It is complicated to compute the centers of the components in feature space. Although the kernel *k*-means algorithm exists [4], it is not considered here because of the high computational complexity. To compute the centers of classes, we use the modified kernel *k*-means method using the distance between the vectors $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ in the feature space [4]:

$$kdist(\mathbf{x}_i,\mathbf{x}_j) = k(\mathbf{x}_i,\mathbf{x}_i) - 2k(\mathbf{x}_i,\mathbf{x}_j) + k(\mathbf{x}_j,\mathbf{x}_j). \quad (13)$$

If we use Gaussian kernel, (13) can be expressed as

$$kdist(\mathbf{x}_i,\mathbf{x}_j) \equiv 2 - 2k(\mathbf{x}_i,\mathbf{x}_j) \quad (14)$$

where the Gaussian kernel is defined as

$$k(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2)) \quad (15)$$

Table 1 shows the modified *k*-means algorithm using $kdist(\cdot,\cdot)$. In (b-2), originally, the real center $\hat{\mathbf{m}}^F_{i,j}$ of the group $\mathbf{G}^F_{i,j}$ in the feature space should be represented by a linear combination of $\mathbf{G}^F_{i,j}$ where $\mathbf{G}^F_{i,j}$ and $\hat{\mathbf{m}}^F_{i,j}$ are the samples belong to *j*th component of *i*th class and the center of $\mathbf{G}^F_{i,j}$ respectively. But this algorithm uses the closest vector from the real center $\hat{\mathbf{m}}^F_{i,j}$. This point is to simplify kernel *k*-means algorithm.

Table 1. Modified kernel *k*-means algorithm using $kdist(\cdot,\cdot)$

```
1. Select k initial centers randomly: $\mathbf{M}^F_i = [\mathbf{m}^F_{i,1},\ldots,\mathbf{m}^F_{i,k}]$
2. Iterate until converged:
(a) E-step:
  Compute $kdist(\cdot,\cdot)$ between $\mathbf{m}^F_{i,j}$ and all of the training
  data in ith class, and assign their closest centers.
(b)M-step:
  for j=1 to k:
    (b-1) Set $\mathbf{G}^F_{i,j}$ to be the group of $n(i,j)$ points whose
    closest center is $\mathbf{m}^F_{i,j}$ (Now the real center $\hat{\mathbf{m}}^F_{i,j}$ of
    $\mathbf{G}^F_{i,j}$ is $\mathbf{G}_{i,j}\mathbf{1}_{n(i,j)\times 1}/n(i,j)$ where $n(i,j)$ is the num-
    ber of samples belonging to $\mathbf{G}^F_{i,j}$)
    (b-2) Set $\mathbf{m}_{i,1}{}^F$ to the closest vector from the real
    center $\hat{\mathbf{m}}_{i,1}{}^F$
```

Using the algorithm in table 1, we can estimate $\mathbf{M}^F_i$ for *i* =1,…,*s*, where *s* is the number of classes (in our experiments, it means speakers). Kernel MFDA is to kenerlize the between-class scatter of MFDA only by using the centers computed by the modified kernel *k*-means. The new kernelized between-class scatter is defined as:

$$\widetilde{\mathbf{S}}_B^F = \sum_{i=1}^{s}\sum_{j=1}^{k}\left(\mathbf{m}_{i,j}^F - \boldsymbol{\mu}^F\right)\left(\mathbf{m}_{i,j}^F - \boldsymbol{\mu}^F\right)' . \tag{16}$$

Let $\mathbf{M}^F = [\mathbf{m}_{1,1}^F,\ldots,\mathbf{m}_{1,k}^F,\ldots,\mathbf{m}_{s,k}^F]$ denote the total set of the centers. Then (16) can be represented by the matrix notation

$$\widetilde{\mathbf{S}}_B^F = (\mathbf{M}^F - \boldsymbol{\mu}^F \mathbf{1}_{1\times sk})(\mathbf{M}^F - \boldsymbol{\mu}^F \mathbf{1}_{1\times sk})' , \tag{17}$$

where $\boldsymbol{\mu}^F$ is computed as

$$\boldsymbol{\mu}^F = \frac{1}{sk}\mathbf{M}^F \mathbf{1}_{sk\times 1} , \tag{18}$$

and plugging (18) into (17), we can derive the following equations

$$\widetilde{\mathbf{S}}_B^F = (\mathbf{M}^F - \frac{1}{sk}\mathbf{M}^F \mathbf{1}_{sk\times sk})(\mathbf{M}^F - \frac{1}{sk}\mathbf{M}^F \mathbf{1}_{sk\times sk})'$$
$$\widetilde{\mathbf{S}}_B^F = \overline{\mathbf{M}}^F \overline{\mathbf{M}}^{F\prime} \text{ where } \overline{\mathbf{M}}^F = (\mathbf{M}^F - \frac{1}{sk}\mathbf{M}^F \mathbf{1}_{sk\times sk}) . \tag{19}$$

To maximize $\widetilde{\mathbf{S}}_B^F$, the eigenvalue decomposition can be applied:

$$\lambda_i \mathbf{u}_i = \widetilde{\mathbf{S}}_B^F \mathbf{u}_i \tag{20}$$

In our method, the following equation is used.

$$\mathbf{u}_i = \overline{\mathbf{R}}^F \boldsymbol{\beta}_i \text{ for } i=1,\ldots \tilde{d} , \tag{21}$$

where $\overline{\mathbf{R}}^F$ is defined to $\mathbf{R}^F - \boldsymbol{\mu}^F \mathbf{1}_{1\times m}$ ($\mathbf{R}^F$ is the reduced set by greedy filtering in section 2.2) and $\boldsymbol{\beta}_i$ is the coefficient of the linear combination of $\overline{\mathbf{R}}^F$. Like kernel PCA in the previous section, by multiplying $\overline{\mathbf{R}}^{F\prime}$ to (20) and plugging-in (19) and (21), the following equation is derived:

$$\lambda_i \overline{\mathbf{R}}^{F\prime}\overline{\mathbf{R}}^F \boldsymbol{\beta}_i = \overline{\mathbf{R}}^{F\prime}\overline{\mathbf{M}}^F\, \overline{\mathbf{M}}^{F\prime}\overline{\mathbf{R}}^F \boldsymbol{\beta}_i . \tag{22}$$

We define $\overline{\mathbf{K}}_{\mathbf{R}} = \overline{\mathbf{R}}^{F\prime}\overline{\mathbf{R}}^F$ and $\overline{\mathbf{K}}_{\mathbf{RM}} = \overline{\mathbf{R}}^{F\prime}\overline{\mathbf{M}}^F$. Then $\overline{\mathbf{M}}^{F\prime}\overline{\mathbf{R}}^F$ is expressed as the transpose of $\overline{\mathbf{K}}_{\mathbf{RM}}$ (i.e. $\overline{\mathbf{K}}_{\mathbf{RM}}'$). Then (22) can be re-written as

$$\lambda_i \boldsymbol{\beta}_i = \overline{\mathbf{K}}_{\mathbf{R}}^{-1}\overline{\mathbf{K}}_{\mathbf{RM}}\overline{\mathbf{K}}_{\mathbf{RM}}'\boldsymbol{\beta}_i \tag{23}$$

where

$$\overline{\mathbf{K}}_{\mathbf{R}} = \mathbf{K}_{\mathbf{R}} - \frac{1}{sk}\mathbf{K}_{\mathbf{RM}}\mathbf{1}_{sk\times m} - \frac{1}{sk}\mathbf{1}_{m\times sk}\mathbf{K}_{\mathbf{MR}} + \frac{1}{(sk)^2}\mathbf{1}_{m\times sk}\mathbf{K}_{\mathbf{M}}\mathbf{1}_{sk\times m} ,$$
$$\overline{\mathbf{K}}_{\mathbf{RM}} = \mathbf{K}_{\mathbf{RM}} - \frac{1}{sk}\mathbf{K}_{\mathbf{RM}}\mathbf{1}_{sk\times sk} - \frac{1}{sk}\mathbf{1}_{m\times sk}\mathbf{K}_{\mathbf{M}} + \frac{1}{(sk)^2}\mathbf{1}_{m\times sk}\mathbf{K}_{\mathbf{M}}\mathbf{1}_{sk\times sk} . \tag{24}$$

Now $\boldsymbol{\beta}_i$ is computed by the eigenvectors of $\overline{\mathbf{K}}_{\mathbf{R}}^{-1}\overline{\mathbf{K}}_{\mathbf{RM}}\overline{\mathbf{K}}_{\mathbf{RM}}'$. Then the eigenvectors corresponding to the $\tilde{d}$ largest eigenvalues are selected:

$$\mathbf{B} = [\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_{\tilde{d}}]. \tag{25}$$

Finally, given the test data $\mathbf{Y} = [\mathbf{y}_1,\ldots,\mathbf{y}_t]$, the transformed data onto kernel $\mathbf{Y}_{output}$ is computed using the kernel matrix $\mathbf{K}_{\mathbf{YR}}$ of $\mathbf{Y}$ and $\mathbf{R}$.

$$\overline{\mathbf{K}}_{\mathbf{YR}} = \mathbf{K}_{\mathbf{YR}} - \frac{1}{sk}\mathbf{K}_{\mathbf{YM}}\mathbf{1}_{sk\times m} - \frac{1}{sk}\mathbf{1}_{t\times sk}\mathbf{K}_{\mathbf{MR}} + \frac{1}{(sk)^2}\mathbf{1}_{t\times sk}\mathbf{K}_{\mathbf{M}}\mathbf{1}_{sk\times m} ,$$

$$\mathbf{Y}_{output} = \mathbf{B}'\overline{\mathbf{K}}_{\mathbf{YR}} \tag{26}$$

Because of the objective of kernel MFDA, the discriminant of the between classes is maximized. Moreover, if we think that the center of each class $\mathbf{M}^F = [\mathbf{m}_{1,1}^F,\ldots,\mathbf{m}_{1,k}^F,\ldots,\mathbf{m}_{s,k}^F]$ represent overall set, it is regarded as kernel PCA of the well sampled set. Especially, like kernel PCA, the test data $\mathbf{Y}$ are normalized by the mean of training set (see (10) and (26)). It is suggestive that the feature vectors can be denoised.

Maximum memory requirements of greedy kernel PCA is $O(lm)$. But our proposed method only needs $O(l/s)$ and $O(sk^2)$ in modified kernel *k*-means algorithm and (24) respectively. In our experiments, the number of samples, classes, and centers per classes in UBM are 330989, 108, and 4 respectively. The number of selected feature vectors by greedy filtering is 100. (i.e. $l = 330989, s = 108, k = 4$ and $m = 100$). Therefore, we only need about 0.5% memory space over greedy kernel PCA.

4. EXPERIMENTAL RESULTS

We used the YOHO database [14] which consists of 138 speakers prompted to read combination lock phrases, for example, “67 34 85”. The YOHO database has ‘enroll’ and ‘verify’ mode. The ‘enroll’ consists of 4 sessions with 24 utterances in each session, and the ‘verify’ consists of 10 sessions with 4 utterances in each session. We used 20 speakers, labeled from 101 to 120 as *true speakers* and 10 speakers, labeled from 121 to 132 as *imposters* and then the rests are used for building a universal background model (UBM). We use only the first session in ‘enroll’ for building speaker models and UBM. And then, all sessions in ‘verify’ for verification task. In the utterances for verification task, we added noises (‘babble’, ‘restaurant’ in Aurora2 database [10]) with SNR 15dB and 20 dB artificially using FaNT[11].

Each frame was represented by 20-dimensional mel-frequency cepstral coefficients with 30ms window and 20ms shift and their energy (21-dimensional feature vectors), and cepstral mean subtraction is applied. The silence was removed automatically using energy threshold. To build the speaker models and universal background model (UBM), we use mixture models [12] with 32 Gaussian components.

We derived the robust features to transform MFCCs using transformation matrix of our proposed method (21th-order vectors are selected). This matrix is derived from the clean utterances used for training UBM. The number of elements m in the reduced set $\mathbf{R}$ is 100. Each speaker in

UBM (108 speakers) is regarded as a class for computing the between class scatter. The number of components (centers) in a class is 4. For kernel function computation, the Gaussian RBF (in (15)) is used with $\sigma^2 = 27661.75$ which is computed by 70 percentile of overall UBM set.

Table 2 shows equal error rates (EER) of the speaker verification system on various environments using various feature vectors. GKPCA means greedy kernel PCA [5][8] and KMFDA means our proposed method. GKPCA outperforms MFCCs and PCA on noisy environments. Our proposed method outperforms overall environments including clean and noisy. The reduced EER are 2.86%, 0.54%, and 0.49% over MFCCs, PCA, and GKPCA respectively. The relative EER are 35.89%, 9.59%, and 8.80% over MFCCs, PCA, and GKPCA respectively.

Table 2. Equal error rate on various conditions (%)

	MFCCs	PCA	GKPCA	KMFDA
Clean	4.66	3.04	3.59	2.74
Babble with SNR 20dB	7.62	6.35	5.71	5.35
Babble with SNR 15dB	10.12	7.59	6.86	6.77
Restaurant with SNR 20 dB	6.88	4.87	4.80	4.37
Restaurant with SNR 15dB	10.52	6.38	7.03	6.29
Average	7.96	5.64	5.59	5.10

5. CONCLUSIONS AND DISCUSSIONS

We have proposed a robust speaker feature extraction method using kernel multimodal Fisher discriminant analysis (kernel MFDA). This method could maximize the discriminant of each class and reduce noises in the features. In experiments, we presented that our proposed method outperforms principal component analysis (PCA) and kernel PCA in various noisy environments.

The parameters to determine in kernel PCA is only σ^2. But kernel MFDA added two parameters more. The first one is how many clusters are in each class. In our proposed method, first of all, the centers are estimated. We have experimented the number of centers in each class with 2~10. Most of cases it outperforms kernel PCA, but the EER of speaker verification was different each case. The second one is what should we regard as classes to discriminate. If our task is speaker identification, it is easy to determine the class. Each class corresponds to each speaker to indentify. However, in speaker verification, it is not that simple. Although we used each speaker in UBM as one class in our experiments, this issue should be considered further. If we have validation set, the overall utterances in UBM can be a class, and the overall utterances of speakers in validation set can be another class for computing (16). These two issues are our future works.

6. REFERENCES

[1] A. Lima, H. Zen, Y. Nankaku, C. Miyajima, K. Tokuda, and T. Kitamura, “On the Use of Kernel PCA for Feature Extraction in Speech Recognition”, In *proc. of eurospeech 2003*, 2003, pp. 2625-2628.

[2] T. Takiguchi and Y. Ariki, “Robust Feature Extraction using Kernel PCA”, In *IEEE int. Conf. on Acoustics, Speech and Signal Processing*, 2006, pp. 509-512.

[3] M.A. El-Gamal, M.F. Abu El-Yazeed, and M.M.H. El Ayadi, “Dimensionality reduction for text-independent speaker identification using Gaussian mixture model”, In *Proceedings of MWSCAS 2003* Vol. 2, 30-30 Dec. 2003, pp. 625- 628.

[4] Shaw-Taylor, J. and Cristianini, N., *Kernel Methods for Pattern Analysis*, Cambridge University Press, New York, 2004

[5] Franc, V. *Optimization Algorithms for Kernel Methods*, PhD thesis, Centre for Machine Perception, Czech Technical University, 2005.

[6] B. Schölkopf, A. Smola, and K.Müller, “Kernel Principal Component Analysis”, In *Int. Conf. on Artificial Neural Networks*, 1997, pp. 583–588.

[7] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, “Fisher discriminant analysis with kernels”, in *Neural Networks for Signal Processing IX*, Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, Eds. Piscataway, NJ: IEEE, 1999, pp. 41–48.

[8] V. Franc and V. Hlavac, “Greedy Algorithm for a Training Set Reduction in Kernel Methods”, *Lecture Notes in Computer Science*, Vol. 2756, 2003, pp. 426-433.

[9] B. Schölkopf, P. Knirsch, and C. Smola and A. Burges, “Fast Approximation of Support Vector Kernel Expansions, and an Interpretation of Clustering as Approximation in Feature Spaces”, In R.-J.Ahler, P.Levi, M.Schanz and F.May, editors, Mustererkennung 1998-20. DAGM-Symp., Berlin, Germany, 1998. Springer-Verlag. pp. 124–132.

[10] H.-G. Hirsch and D. Pearce, “The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions”, in *ASR 2000*, 2000, pp. 181–188.

[11] H.-G. Hirsch, “Fant-fltering and noise adding tool”, http://dnt.kr.hs-niederrhein.de/download.html.

[12] D. Reynolds, “Speaker identification and verification using Gaussian mixture speaker models”, *Speech Communication*, vol. 17, 1995, pp. 91–108.

[13] M.-S. Kim, I.-H. Yang and H.-J. Yu, “Robust Speaker Identification Using Greedy Kernel PCA”, In *Proc. of 20th IEEE International Conference on Tools with Artificial Intelligence*, 2008, pp. 143-146.

[14] Campbell, J. P., Reynolds, D. A., “Corpora for the Evaluation of Speaker Recognition Systems.” In *IEEE int. Conf. on Acoustics, Speech and Signal Processing*, 1999, pp. 829–832.