

Gaussian Selection Using Self-Organizing Map for Automatic Speech Recognition

Yujun WANG, Hugo Van hamme

ESAT Department, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10,
B-3001 Leuven, Belgium
{yujun.wang, hugo.vanhamme}@esat.kuleuven.be

Abstract. The Self-Organizing Map (SOM) is widely applied for data clustering and visualization. In this paper, it is used to cluster Gaussians within the Hidden Markov Model (HMM) of the acoustic model for automatic speech recognition. The distance metric, neuron updating and map initialization of the SOM are adapted for the clustering of Gaussians. The neurons in the resulting map act as Gaussian clusters, which are used for Gaussian selection in the recognition phase to speed up the recognizer. Experimental results show that the recognition accuracy is kept while the decoding time can be reduced by 70%.

Keywords: SOM, Speech recognition, Gaussian clustering, Gaussian selection.

1 Introduction

The Self-Organizing Map (SOM) is a widely applied neural model for data analysis and especially for clustering. Its algorithms are comprehensively formulated in [1]. The SOM already attracted interests of the researcher in speech recognition as a vector quantization method to classify speech features, e.g. [20] and [21]. Research hat used the SOM to cluster the Hidden Markov Models (HMM) in speech recognition is described in [15]. The author directly treated the parameters of the HMMs as the input features to the SOM. In this paper, SOM is used for clustering Gaussians of the acoustic model for automatic speech recognition.

In a HMM based state-of-the-art Large Vocabulary Continuous Speech Recognition (LVCSR) system [6], there are typically over twenty thousand Gaussians. During the decoding phase, i.e. at recognition time, all the Gaussians need to be evaluated given a 39 dimensional observation vector, which is renewed every 10 milliseconds. Hence evaluation of Gaussians is one of the most time-consuming tasks for the recognizer. However, given the observed feature vector, only a small subset of Gaussians dominate the likelihood of the states of the HMM, while the rest are unlikely. To speed up the decoding, vector quantization based Gaussian selection ([7] [10] [11]) was proposed to exclude unlikely Gaussians from evaluation. Here, cluster Gaussians are computed and assigned likelihoods by the decoder. Only the member Gaussians belonging to those likely clusters are evaluated. The clustering method in the previous research is hard K-Means. SOM, as a soft clustering technique, is closely

related to K-Means [3]. It is formed of neurons located on a regular, usually low dimensional grid. The neurons are connected to adjacent neurons usually by a Gaussian neighborhood function preserving the topological properties of the input space. The weights of each input training vector (in our case an input Gaussian) for updating the neurons or codebook is determined by the neighborhood function and the map topology. The output neurons then define cluster Gaussians which can be assigned likelihoods, hence indicating if their cluster members are likely to score well in their turn and if spending computer time on their evaluation is useful or not.

Figure 1 shows an example of the unified distance matrix [4] of a 2-dimensional hexagonal SOM of Gaussians. The matrix visualizes the distance between adjacent neurons. A dark coloring corresponds to a large distance. The distance metric between neurons will be explained in the next section. The input Gaussian and the output neurons are expressed in the spectral domain. The connected gray and black cells form the boundaries of distinct regions on the map. Sample spectra of the vowel /ɑ:/, /i:/, /u:/, /ɔ:/, /æ/ in IPA are plotted on the map. While the mapping serves the primary goal of clustering of Gaussians in this paper, it also has a diagnostic value in engineering acoustic models. Furthermore, research such as [19] claims that the SOM provides an interaction to assist speaker to optimize their pronunciation.

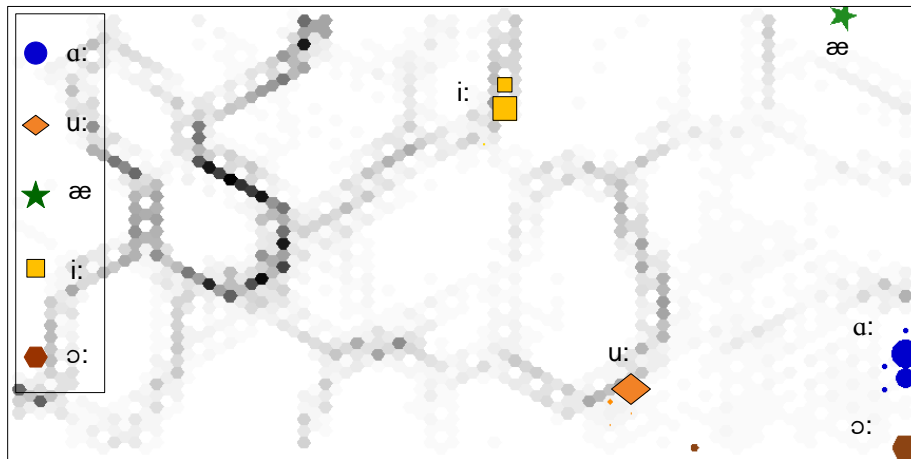


Fig. 1. Unified distance matrix [4] of a SOM plotted by SOM Toolbox [18]. The input Gaussians and output neurons are expressed in the spectral domain. Sample spectra of five vowels are plotted on the map: Given a frame of spectral features, the likelihood of every neuron is calculated and the one with highest likelihood is marked as the best matching unit of the feature vector. The size of the markers indicates the hit rate of the spectral features on a particular neuron.

The rest of the paper is organized as follows: section 2 introduces the SOM training procedure and the adaptations to the algorithms to order Gaussians; section 3 describes our scheme of Gaussian selection using the SOM. Section 4 shows the experimental settings and results. In the last section, conclusions are drawn and future work is proposed.

2 Training a SOM of Gaussians

The SOM in our work has a two-dimensional hexagonal topology and is trained in a batch mode, i.e. updating all the neurons or clusters after presenting all training data. Our training data are the N Gaussians of the acoustic model of the speech recognizer, henceforth called *member Gaussians*. Further suppose there are M neurons. The process for the batch ordering is:

- 1 Initialize the map to a principal linear sub-manifold.
- 2 For training episode t from 1 to T
 - For the n^{th} member Gaussian, n from 1 to N
 - 2.a Find its best matching neuron $c(n)$
 - 2.b For the m^{th} neuron, m from 1 to M
 - Calculate the neighborhood function $g(n, m, t)$
 - 2.c Update the codebook

The neighborhood function in step 2.b has the expression:

$$g(n, m, t) = \exp\left(-\frac{\|r_{c(n)} - r_m\|^2}{2\beta^2(t)}\right) \quad (1)$$

where r_m denotes the coordinates of the m^{th} neuron on the SOM, $\beta(t)$ is the neighborhood radius which decays with the training episode t .

The algorithm within each step in the above pseudo code should be adapted to handle the self organizing of Gaussians. Algorithm adaptations for map initialization in step 1, the distance metric between the input training data (member Gaussians) and the output neurons (cluster Gaussians) in step 2.a and the codebook estimation in step 2.c will be covered in the section 2.1 and 2.2.

2.1 Distance Metric and Neuron Estimation

The Symmetric Kullback-Leibler Divergence (SKLD) is commonly used to measure the distance between a particular input member Gaussian and every neuron in step 2.a. If p and q are multivariate Gaussians, their SKLD is:

$$\text{SKLD}(p, q) = \frac{1}{2} \text{trace}\left\{(\Sigma_p^{-1} + \Sigma_q^{-1})(\mu_p + \mu_q)(\mu_p + \mu_q)' + \Sigma_p \Sigma_q^{-1} + \Sigma_q \Sigma_p^{-1} - 2\mathbf{I}\right\} \quad (2)$$

The estimation of neurons is based on the approach in [9], where a method for finding the centroid of a set of Gaussians is derived. In their work, the centroid is the Gaussian that minimizes the sum of SKLD to each of the set members. In our work, we extend the results of [9] by minimizing the weighted within-cluster mean SKLD for the m^{th} neuron:

$$\overline{SKLD}_m = \frac{\sum_{n=1}^N SKLD(n,m) \cdot g(n,m,t)}{\sum_{n=1}^N g(n,m,t)} \quad (3)$$

In step 2.b, equation (3) is minimized given the N member Gaussians and their corresponding weights, $g(n,m,t)$, to update one of the M neurons. Hence the formula to re-estimate the mean of the m^{th} neuron is:

$$\bar{\boldsymbol{\mu}}_m^t = \left[\sum_{n=1}^N g(n,m,t) (\boldsymbol{\Sigma}_n^{-1} + \boldsymbol{\Sigma}_m^{-1}) \right]^{-1} \left[\sum_{n=1}^N g(n,m,t) (\boldsymbol{\Sigma}_n^{-1} + \boldsymbol{\Sigma}_m^{-1}) \boldsymbol{\mu}_n \right] \quad (4)$$

Matrix \mathbf{B} is constructed to facilitate the re-estimation of the covariance matrices for the neurons:

$$\mathbf{B} = \begin{bmatrix} 0 & \mathbf{A} \\ \mathbf{C} & 0 \end{bmatrix} \quad (5)$$

where

$$\mathbf{A} = \sum_{n=1}^N g(n,m,t) [(\boldsymbol{\mu}_n - \bar{\boldsymbol{\mu}}_m^t)(\boldsymbol{\mu}_n - \bar{\boldsymbol{\mu}}_m^t)' + \boldsymbol{\Sigma}_n] \quad (6)$$

and

$$\mathbf{C} = \sum_{n=1}^N g(n,m,t) \boldsymbol{\Sigma}_n^{-1} \quad (7)$$

Suppose the member Gaussians are d -dimensional, then \mathbf{B} has d positive and d symmetrically negative eigenvalues. Then a $2d$ by d matrix \mathbf{V} is constructed whose columns are the d eigenvectors corresponding to the positive eigenvalues. \mathbf{V} is partitioned in its upper halve \mathbf{U} and lower halve \mathbf{W} :

$$\mathbf{V} = \begin{bmatrix} \mathbf{U} \\ \mathbf{W} \end{bmatrix} \quad (8)$$

$$\bar{\boldsymbol{\Sigma}}_m^t = \mathbf{U}\mathbf{W}^{-1} \quad (9)$$

It can be seen from equation (4) and (6) that the procedure of estimating the neurons given the weights $g(n,m,t)$ is iterative. The calculation of the mean depends on the previously calculated covariance and vice versa. The exit criterion is the convergence of mean SKLD defined in equation (3). The choice of the initial values is introduced in section 2.2.

2.2 Map Initialization

The purpose of step 1 in the pseudo code is to obtain faster or even better ordering convergence. A global Gaussian (or a single-neuron map) is calculated by averaging the entire set of member Gaussians. Then the mean and covariance matrix of the global Gaussian are updated using equation (2) to (9) for several iterations till the mean SKLD in equation (3) is converged. Principal Component Analysis (PCA) is applied on the covariance matrix to find the first two principal eigenvectors e_1 , e_2 and eigenvalues, λ_1 , λ_2 . The square roots of the two principal eigenvalues are used to determine the height h and width l whose product is the size of the code book, M . Then the map is spanned linearly along the directions of the two principal components, i.e. the means of the $(x,y)^{\text{th}}$ neurons are initialized as $(r_x l / \sqrt{\lambda_2}) e_2 + (r_y h / \sqrt{\lambda_1}) e_1$, where r_x and r_y are the hexagonal coordinates on the map. The initial values of the covariance matrices are simply assigned with the average values over all member Gaussians' covariance matrices.

3 VQ Based Gaussian Selection Using SOM

Evaluation of Gaussians is one of the computationally intensive tasks in an HMM based LVCSR system. It typically consumes 40% to 70% of the total recognition time without any pruning approaches. Many methods of Gaussian selection emerged to reduce the number of Gaussians to be calculated during decoding. They can be classified as axis indexing based methods and VQ-based methods ([10] [11]). The former quickly locates the observation based on the indices which are already created in the training phase then decide which Gaussian is selected [13] or removed [14]. The latter evaluates the cluster Gaussians and selects only the member Gaussians belonging to those cluster Gaussians having high likelihoods during decoding [10] [11]. Other than Gaussian selection, the VQ based techniques can also be used for sub-space Gaussian clustering [8], which is a different idea to speed up the recognizer.

Our SOM-based Gaussian selection method utilizes the VQ-based methods. Whether a particular member Gaussian is selected is determined by a short list of neurons and the neuron-member Gaussian mapping table, as explained below.

3.1 Constructing the Short List of Neuron Selection

The motivation of Gaussian selection is that only a small portion of Gaussians dominate the likelihoods of the states of the HMM per frame, and are worth evaluating. The SOM consisting of connected neuron Gaussians preserves the topology of the input Gaussian space, i.e. only small regions of neurons on the map dominate the likelihoods of the entire map. In practice, around 80% of the neurons are negligible due to their close-to-zero likelihoods. The posterior probabilities of the remaining 20% neurons are calculated from their likelihoods and sorted. The list is

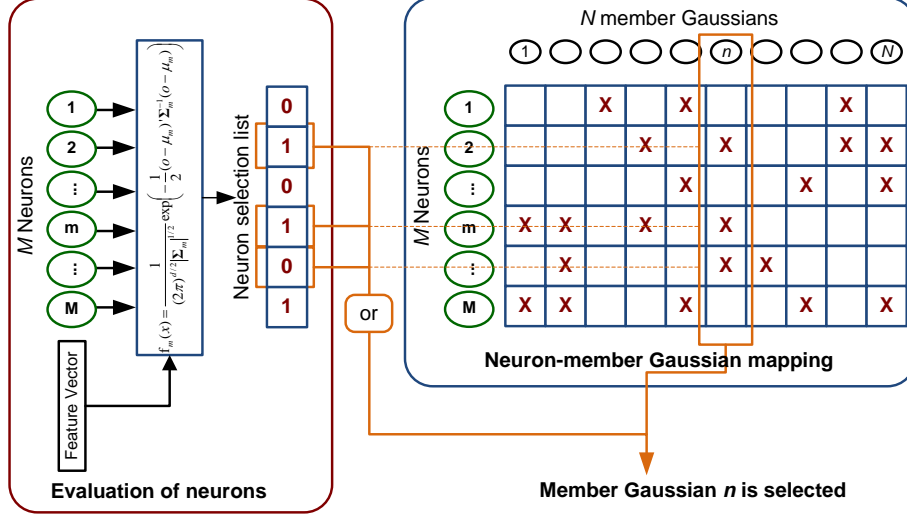


Fig. 2. Decision on the selection of a member Gaussian. During decoding, whether a member Gaussian is selected is determined by the neuron-member Gaussian mapping table and the neuron selection list (“1” indicates that the corresponding Gaussian is selected). The mapping table determines to which neurons the member Gaussian belongs. The recognizer then checks the mapping table and will evaluate the member Gaussian only if any of those neurons is selected.

then truncated further to the length such that 95% of the posterior probability mass is included, i.e. the length of the short list is the smallest j such that:

$$\sum_{k=1}^j p(G_k | O)^\alpha > 0.95 \sum_{m=1}^{M^{*0.2}} p(G_m | O)^\alpha \quad \text{where} \quad p(G_k | O) \geq p(G_{k+1} | O) \quad (10)$$

where the exponent α is introduced to compensate for unmodeled correlations and will indirectly control the number of selected Gaussians. $p(G_k|O)$ denotes the posterior probability of cluster Gaussian k . In figure 2, the neurons labeled by “1” in the neuron selection table appear in the short list.

3.2 Mapping Member Gaussians to Neurons

The neuron-member Gaussian mapping table in figure 2 carries the “softness” of the SOM clustering. Each member Gaussian can be assigned to maximally 6 neurons, namely the first through 6th best matching units on the map. The neurons with SKLD greater than the multiplication of a pruning factor θ and the SKLD of the best matching unit are pruned from the mapping table. On average 3.6 neurons are retained for each member Gaussian. As shown in figure 2, a member Gaussian is selected if at least one of the neurons to which it belongs is activated in the neuron selection table.

Keeping a single neuron Gaussian per member Gaussian in the Gaussian mapping table certainly excludes more member Gaussians from being calculated, but it is

found to degrade the recognition accuracy. The result is listed in table 1 as Single Mapping SOM.

4 Experiments

Speech recognition experiments were conducted on the Aurora4 [12] large vocabulary database, which is derived from the WSJ0 Wall Street Journal 5k-word dictation task. The test set is the clean-condition subset containing 330 utterances from 8 different speakers.

4.1 Experiment Settings

The acoustic model is trained using the clean-condition training set which contains 7138 utterances from 83 speakers, which is equivalent to 14 hours of speech data. All recordings are made with the close talking microphone and no noise is added. The speech spectra, its first and second order derivatives are transformed into 39 dimensional MIDA (Mutual Information based Discriminant Analysis [16]) features and further decorrelated to ensure the model can use diagonal covariance Gaussians. There are 4091 tied states, or senones, and they share 21087 Gaussians. A bigram language model for a 5k-word closed vocabulary is provided by Lincoln Laboratory. The decoding is done with a time-synchronous beam search algorithm and the detail can be found at [17]. The recognizer was launched on a PC installed with Dual Core AMD Opteron Processor 280, whose main frequency is 2.4 GHz. Only one core is activated for the testing.

4.2 The SOM

The two dimensional hexagonal SOM is trained using SOM Toolbox [18] with the 21087 MIDA diagonal covariance Gaussians as input. The map size is 26 by 20, i.e. 520 neuron Gaussians are in the map. The covariance matrices of the output neurons are constrained to be diagonal as well. A rough training phase with only 6 iterations of ordering is carried out first to prevent the map from topological defects [5], then followed by a fine ordering with 24 iterations.

Though the convergence of the typical SOM is not strictly proved in higher-than-one-dimensional case theoretically [5], the mean SKLD is observed monotonously decreased during ordering process in the experiment.

4.3 Experiments Using Other Approaches

Two additional approaches, namely the K-Means and a HMM-based method, are implemented to compare with the SOM.

K-Means uses the following cost function:

$$f_{K-Means} = \sum_{n=1}^N \left(\sum_{m=1}^M w_{nm} SKLD(n, m) + \gamma \sum_{m=1}^M w_{nm} \log \frac{1}{w_{nm}} \right) \quad (11)$$

The weight to update cluster m using member n , w_{nm} , is

$$\begin{aligned} w_{nm} &= \arg \min_{\sum_{m=1}^M w_{nm}=1} \left(\sum_{m=1}^M w_{nm} SKLD(n, m) + \gamma \sum_{m=1}^M w_{nm} \log \frac{1}{w_{nm}} \right) \quad (12) \\ &= \exp(-SKLD(n, m) / \gamma) / \sum_{i=1}^M \exp(-SKLD(n, i) / \gamma) \end{aligned}$$

Here the softness of the clustering is controlled by γ . The number of clusters M is 520.

An alternative approach to acquire clusters is to train them using the data. A compact HMM containing 520 Gaussians is trained using the same training data as the full HMM containing the member Gaussians. It shares the same structure of tied states as the full model. These 520 Gaussians are used as the cluster Gaussians. An extra Viterbi segmentation pass after the training is carried out to set up the association table between the member Gaussians and the cluster Gaussians: each speech frame is hence assigned to a HMM state. The association count between the dominating member Gaussians of that HMM state and the dominating cluster Gaussian is then incremented by 1. This association table is used as the cluster-member Gaussian mapping table after a proper truncation, i.e. per member Gaussian keeping only the cluster Gaussians with the highest association count (3.6 cluster Gaussians on average).

4.4 Experimental Results

Table 1 shows the experimental results of the Gaussian selection using different approaches of Gaussian clustering, which are compared with the baseline system where none of the Gaussians are pruned during decoding. The percentage of calculated Gaussians is the ratio between the number of calculated Gaussians (including both neurons and selected member Gaussians) and 21087. The baseline is a 2.2×realtime system. We achieve a 0.67×realtime system using the SOM, thus 70% recognition time is saved while the Word Error Rate (WER) is even lower than the base line. The K-Means approach yields lower WER than the SOM, but calculates more Gaussians. The single mapping SOM, where only one neuron Gaussian is kept per member Gaussian in the mapping table, loses accuracy while reducing 1.2ms CPU time per frame, is not preferable. The HMM-based method cannot improve the performance but is slower than the SOM. The associated indexing based approach of Gaussian selection, called Fast Removal of Gaussians (FRoG) [14], of the recognizer [17] is also tested. It only calculated about 5% of the member Gaussians, but required 1.2×realtime.

The Gaussian selection systems based on the SOM, K-Means and data-driven approach also helps to reduce the beam search time because it removes unlikely

Gaussians which dominate the unlikely states, hence the confusion among the different search paths is reduced.

Table 1. Word Error Rates and CPU time of SOM Gaussian selection on Aurora 4

| | WER | CPU Time | | %Gaussian calculated |
|-----------------------|-------|---|------------------------|----------------------|
| | | Gaussian Calculation (ms/frame) | Beam Search (ms/frame) | |
| SOM | 6.76% | 1.8 | 4.9 | 7% |
| Single Mapping SOM | 7.02% | 1.4 | 4.1 | 4% |
| K-Means | 6.65% | 2.4 | 5.1 | 11% |
| Data driven | 6.82% | 2.2 | 5.2 | 9% |
| FRoG | 6.82% | The CPU time per frame is 12ms in total | | ≈5% |
| No Gaussian Selection | 6.87% | 15.3 | 6.7 | 100% |

5 Discussion and Future Work

SOM algorithms which are capable of ordering and clustering Gaussians are implemented, tested and proved valid by experimentation. They showed effective to selectively evaluate Gaussians in an automatic speech recognizer, resulting in competitive speed-ups.

The SOM algorithm we used can be interpreted as a soft K-Means with decreasing learning rate. As both K-Means and SOM have a multitude of variants, and the difference of the performance of the SOM and K-Means in our experiments are marginal, it is hard to decide which method is superior. K-Means and SOM can be the substitute of each other. However, SOM provides an insightful visualization of the space of Gaussians, which can facilitate analyzing and engineering the acoustic models for speech recognition.

In our SOM-based Gaussian selection, the evaluation of neurons is inevitable. There are few hundreds of neurons that need to be evaluated additionally over the member Gaussians. However, not all of these evaluations are worthwhile, because only small regions of neurons in the short list are likely. Watanabe et al. [10] presented a tree structure of Gaussians (i.e. clusters of clusters) where the search for likely clusters is narrowed down at the top levels of the tree, hence it can reduce the number of cluster Gaussians to be calculated. A similar idea could be implemented using the hierarchical SOM (also known as the growing SOM or GSOM) [2][21] to quickly locate the likely regions hierarchically. This could be an extension of our work in the future.

Acknowledgments. This research is financed by the MIDAS project of the Nederlandse Taalunie under the STEVIN programme.

References

1. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer-Verlag, Berlin (2001)

2. Rauber, A., Merkl, D., Dittenbach, M.: The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High-Dimensional Data. In: IEEE Transactions on Neural Networks, vol. 13, pp. 1331-1341. (2002)
3. Sueli, A. M., Joab O. L., M.: Comparing SOM neural network with Fuzzy c-means, K-means and traditional hierarchical clustering algorithms. In: European Journal of Operational Research, pp. 1742-1759 (2006)
4. Ultsch, A. Siemon, H.P.: Kohonen's self organizing feature maps for exploratory data analysis. In: Proceeding of International. Neural Network Conference, pp. 305-308, Dordrecht, Netherlands (1990)
5. Van Hulle, M.M.: Faithful Representations and Topographic Maps: From Distortion- to Information-Based Self-Organization. John Wiley & Sons, New York (2000)
6. Huang X., Hon, H.W., Reddy, R.: Spoken Language Processing: A Guide to Theory, Algorithm, and System Development. Prentice Hall, NJ (2001)
7. Bocchieri, E.: Vector quantization for efficient computation of continuous density likelihoods. In: Proceeding of ICASSP, vol 2, pp. 692-695 (1993)
8. Bocchieri, E., Mak, B.K.-W.: Subspace Distribution Clustering Hidden Markov Model. IEEE Transactions on Speech and Audio Processing, vol 9, pp.264-275 (2001)
9. Myrvoll, T.A., Soong, F.K.: Optimal Clustering of Multivariate Normal Distributions Using Divergence and Its Application to HMM Adaptation. Proceeding of ICASSP, pp. 552-555 (2003)
10. Watanabe, T., Shinoda, K., Takagi, K., Iso, K.-I.: High Speed Speech Recognition Using Tree-Structured Probability Density Function. Proceeding of ICASSP, vol 1, pp. 556 – 559 (1995)
11. Shinoda, K., Lee, C.-H.: A structural Bayes approach to speaker adaptation. IEEE Transactions on Speech and Audio Processing, vol 9, pp. 276-287 (2000)
12. Parihar, N., Picone, J.: An Analysis of the Aurora Large Vocabulary Evaluation. Proceeding of Eurospeech, pp. 337-340 (2003)
13. Fritsch, J., Rogina, I.: The Bucket Box Intersection (BBI) Algorithm for Fast Approximate Evaluation of Diagonal Mixture Gaussians. In Proceeding of ICASSP, vol 2, pp. 273-276, Atlanta (1996)
14. Demuynck, K.: Extracting, Modeling and Combining Information in Speech Recognition. PhD thesis, K.U.Leuven, ESAT (2001.)
15. Du, X.-P., He, P.-L.: The Clustering Solution of Speech Recognition Models with SOM. Advances in Neural Networks – ISNN 2006, Lecture Notes in Computer Science, pp. 150-157 (2006)
16. Duchateau, J., Demuynck, K., Van Compernelle, D., Wambacq, P.: Class definition in discriminant feature analysis. In: Proceeding of European Conference on Speech Communication and Technology, pp. 1621-1624, Aalborg, Denmark (2001).
17. SPRAAK. Speech Processing, Recognition and Automatic Annotation Kit, <http://www.spraak.org/>
18. SOM Toolbox, <http://www.cis.hut.fi/somtoolbox>
19. Wang, X., Xue, L., Yang, D., Han, Z.: Speech Visualization based on Robust Self-organizing Map (RSOM) for the Hearing Impaired. In: Proceeding of International Conference on BioMedical Engineering and Informatics, pp. 506-509 (2008)
20. Sarma, M.P., Sarma, K.K.: Speech Recognition of Assamese Numerals Using Combinations of LPC - Features and Heterogenous ANNs. In: Proceeding of International Conference on BioMedical Engineering and Informatics, pp. 506-509 (2008)
21. Souza Júnior, A.H., Barreto, G.A., Varela, A. T.: A speech recognition system for embedded applications using the SOM and TS-SOM networks. In: J. I. Mwasiagi. (Org.). Self Organizing Maps: Applications and Novel Algorithm Design. Viena, Austria: IN-TECH publishing, pp. 97-108 (2010)