A NOVEL FEATURE EXTRACTION METHOD FOR ISOLATED WORD RECOGNITION BASED ON NESTED TEMPORAL AVERAGING

Radu Dogaru, Gabriel Nicolae Costache, Octavian Dumitru, and Inge Gavat

Department of Applied Electronics and Information Engineering, Polytechnic University of Bucharest, Romania, e-mail: radu_d@ieee.org,

Abstract: A novel preprocessing method is proposed. It has a reduced complexity and therefore is aimed to be used in low power, VLSI implemented, speech recognizers. Our algorithm extracts a feature vector made from up to 3 feature vectors, each coming from a particular variable length speech sequence. The sequences are nested one into each other while their length is divided by 2 for each nesting operation. Each feature vector is computed as an average, min and max of all 13-dimensional Mel-cepstral coefficients obtained within a sound sequence. On a sound database with 10 speakers speaking 7 different words the classification performance was found to be close and even better than the one obtained using traditional methods (HMMs).

Keywords: Speech recognition, Pattern classification, Neural Networks, Support Vector Machines.

1. INTRODUCTION

There are various circumstances where a compact and low power device for isolated word recognition is required (e.g. input device for persons with motor disabilities, mobile devices such as PDAs or telephones, etc.). For the case of isolated word a sequence of variable length can be generated. Typically the isolated word speech recognition process is based on the HMM (Hidden Markov Model) method (Rabiner, 1989). The HMM method has the advantage that it does not depend on the specific word length and that it considers the change from successive feature vectors coming from the front-end processor, which encodes important information to recognize properly the words. But on the other hand the HMM method is computationally intensive. On the other hand, the use of neural networks or other machine-learning algorithms (e.g. support vector machines, or SVM (Vapnik, 1999)) may drastically simplify the hardware requirements but it requires a *fixed length feature vector*, which contradicts with the variable length nature of the isolated word signals. In the following we propose an easy to implement preprocessing method. Its performances in terms of percentage of correctly classified words appears to be superior of the method using discrete HMMs on the same database. Section 2 briefly presents the Mel-cepstral front end used to extract spectral information. The novel preprocessing method called a Nested Temporal Averaging (NTA) is introduced in Section 3, while results on a benchmark database are presented in Section 4. Conclusions are discussed in section 5.

2. AUDIO SIGNAL PROCESSING

The first step in all recognition or classification tasks is signal analysis, where the signal is processed in order to obtain the important characteristics, further

This paper was recommended for publication by Laurențiu Frangu

called features or parameters. By using only the important characteristics of the signal, the amount of data used for comparisons is greatly reduced and thus, less computation and less time is needed for comparisons.

Our audio parameter extraction is based on perceptual linear predictive coding and perceptual cepstral coding, methods. In the next we will focus mostly on the cepstral coding method, since it is widely accepted as a robust voice preprocessing method. The block scheme of the processor is shown in the Fig 1.

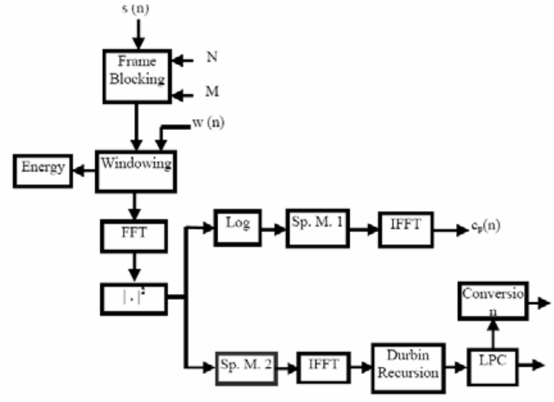

Fig.1. . Acoustic processor

Few blocks are common for both linear prediction and cepstral coding. The first block in the scheme (frame blocking) is necessary because of the non-stationary character of speech signals. Its role is to extract short fragments of the speech signal, called *frames*, to be further processed. During each frame, speech is approximated as a quasi-stationary random process.

Then we put each frame through a Hamming window. We can compute at this time the energy of each frame and we can use the energy set of coefficients in the recognition process for more accuracy.

Cepstral analysis is a very reliable method to audio parameterization and can be realized applying the blocked and windowed time discrete signal $x(n)$ to the processing chain depicted in Figure 1. After the Discrete Fourier Transform (DFT), the modulus of the signal is calculated and the logarithm is taken, the result being proportional in fact to the power spectrum of the speech signal. Through IDFT, the real cepstrum is obtained, the “filter” characterization being comprised near of the cepstrum origin. The resampling of the real cepstrum leads to the cepstral coefficients, which, alone or in addition with the energy E, and/or the first and second order differences constitute a feature vector successfully applied in speech recognition.

In order to obtain a parametric representation with the mel-cepstral coefficients and their first and second order variations the power spectrum is processed using a set of filters (denoted Sp.M.1 in Fig.1) with the transfer functions represented in the figure2.

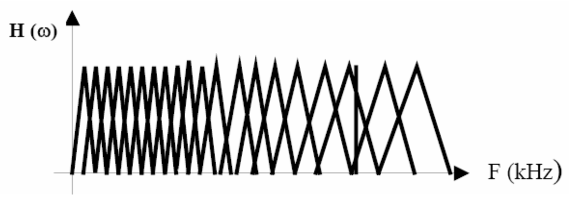

Fig.2. The transfer functions of Mel filters

That filter bank is a model for the critical band perception of the human cochlea. The outputs of the filters (frequency domain) are calculated by the formula (1), where *i* is the index number of the Mel filter, S is the discrete FFT transform of the input signal *s(t)*, *k* is the integer index of the input signal frequency space.

$$(1)\quad Y(i) = \sum_{k=0}^{N/2} \log\left(\left|S(k)\right| H_i\left(\frac{2\pi k}{N} \right) \right)$$

DCT is then applied to the resulted vector Y to compute the Mel-cepstral coefficients used as features for speech recognition. A more detailed description of the Mel-cepstral front end is beyond the focus of this paper. The interested reader may find more details in (Binu and Mathew, 2004).

3. THE NESTED TEMPORAL AVERAGE (NTA) ALGORITHM

A sequence of voice samples representing a spoken word is transformed as above in a corresponding sequence of cepstral parameter vectors, each vector being generated from the samples within a window of the speech signal. The quantity of information represented by all these vectors is still too large to be directly applied to a classifier. On the other hand the number of parameters is variable, while the number of inputs to a classifier is usually fixed. Instead of employing a HMM model for the word recognition, we propose a simple and effective method for transforming a *sequence of any length* into a *fixed sized feature vector* containing the significant features of the uttered sound. For reasons that will be clear later, we call this method a Nested Temporal Averaging (NTA) method.

The variable length sequence $\{\mathbf{c}_1,..\mathbf{c}_j,..\mathbf{c}_L\}$ of voice parameter vectors corresponding to an isolated word is transformed by NTA into a fixed size feature vector with up to *6n* components. The word length (number of samples) *L* is variable and it depends on the particular sound (word) uttered. We assume that words are already segmented.

Within this paper, each parameter vector $\mathbf{c}_j$ is associated with the n=13 Mel-cepstral coefficients computed as described previously on a window of $N=256$ consecutive speech samples, i.e. $\mathbf{c}_j = \left[c_1^j,..c_i^j,...c_n^j\right]$. The NTA method can be applied as well to other types of speech parameter vectors.

The first step in our algorithm assumes *scaling* the values of the speech parameter components such that $c_i^j \in [-1,1]$.

Scaling is achieved using the following simple rule: For each „channel“ *i*, and for the entire available dataset *B* one shall compute $M_i = \max_{j \in B}(C_i^j)$ and $m_i = \max_{j \in B}(C_i^j)$, where C_i^j is the raw value of the voice parameter *i* for sample *j*. Then compute: $c_i^j = -1 + 2\dfrac{C_i^j - m_i}{M_i - m_i}$.

The following *reduced voice parameters* can be now computed as follows:

1) $\mathbf{f}_1 = \left[mean_1,..mean_i,..mean_n\right]$,

where $mean_i = \dfrac{1}{L}\sum_{j=1}^{L} c_i^j$

This feature vector extracts the average features of the voice parameters composing the original sequence and it does not depend on *L*.

2) $\mathbf{f}_2 = \left[\min_1,..\min_i,..\min_n\right]$ where $\min_i = \min_{j=1,..L}\left(c_i^j\right)$

3) $\mathbf{f}_3 = \left[\max_1,..\max_i,..\max_n\right]$ where $\max_i = \max_{j=1,..L}\left(c_i^j\right)$

A similar set of *reduced voice parameter vectors* $\mathbf{f}_1', \mathbf{f}_2', \mathbf{f}_3'$ can be obtained applying the above procedures to a reduced sequence $\left\{\mathbf{c}_1',..\mathbf{c}_j',..\mathbf{c}_L'\right\}$ nested within the original sequence. The reduced sequence contains half of the vectors from the original sequence as follows:

$\mathbf{c}_j' = \mathbf{c}_{j+\lfloor L/4 \rfloor}$, when $j = 1,.., L'$ and $L' = \left\lfloor \dfrac{L}{2} \right\rfloor$.

Here $\lfloor x \rfloor$ is the lowest integer close to *x*.

The features computed on the first-order nested sequence were introduced to include that information which is of temporal nature and which is important for an accurate sound recognition. Indeed it is easy to prove that if a random permutation is applied to the vectors from the initial sequence, the vectors from the reduced sequence are different and therefore will determine different values of the $\mathbf{f}_1', \mathbf{f}_2', \mathbf{f}_3'$ features.

The above procedure can be recursively applied, producing another reduced (nested) sequence from the first-order nested one, and so on up to $\log_2(L)$ levels. However in practice it turned out that the first two levels (the original and the first-order reduced sequence) suffice and additional levels do not improve significantly the classification performance, while the computational complexity is increased. A graphic representation of the NTA algorithm is depicted in Fig.3.

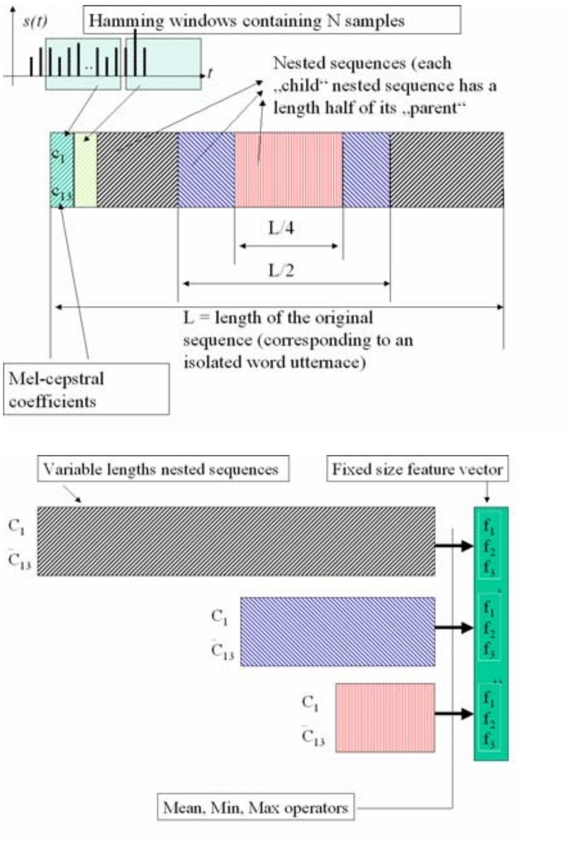
s(t)
Hamming windows containing N samples
t
Nested sequences (each „child“ nested sequence has a length half of its „parent“
c1
c13
L/4
L/2
L = length of the original sequence (corresponding to an isolated word utternace)
Mel-cepstral coefficients
Variable lengths nested sequences
Fixed size feature vector
C1
C13
f1
f2
f3
Mean, Min, Max operators

Fig.3. A graphic sketch of the NTA preprocessing algorithm

Note than all mathematical operators involved in the above algorithms are simple (addition and comparisons) and the computational complexity in computing the features is $O(n \cdot L)$. For instance, using a typical value L=60 and n=13 in order to calculate a feature vector $\mathbf{F4} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_1' \mathbf{f}_2', \mathbf{f}_3']$ with 6 components, some 6*70*13=5460 arithmetic computations are necessary. On the other hand, it is easy to demonstrate that the computational complexity of the SVM (Vapnik, 1999) used in a recognition phase (assuming that it has been trained offline previously) is $O(n \cdot Sv)$ where M (the number of classes) is included in SV (i.e. the number of support vectors). These values compare favorably with the complexity of the HMM algorithm under similar circumstances.

The minimal choice for a feature vector associated with a word is $\mathbf{F} = [\mathbf{f}_1]$ (i.e. n components) while a maximal feature vector considered herein is $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_1', \mathbf{f}_2', \mathbf{f}_3']$ (i.e. *6n* components). As seen in the following, the classification performances improve and reach a stationary value for the maximal feature vector.

4. SVM CLASSIFIERS

The feature vector $\mathbf{F}$ determined at the output of the NTA preprocessing algorithm can be applied to a large variety of classifiers. NTA is capable to provide a fixed length, reasonable sized feature vector F associated with a spoken word. It can be easily classified with a variety of neural classifiers (Haykin, 1999), many of them having convenient VLSI implementation. Our choice here was for the Support Vector Machine (Vapnik, 1999), a classifier deeply rooted in the statistical machine learning theory, proved so far to have best classification performances when compared with other neural classifiers. In our simulations we use the multi-class SVM implementation available from (Junshui et al., 2003). For a detailed tutorial on SVM the reader is pointed to (Hearst et al., 1998).

5. EXPERIMENTAL RESULTS

We considered a database from (Audio-database, 2001) reduced to M=9 classes (i.e. the utterances of the figures "two", "three", "ten") uttered by 10 speakers (3 female and 7 male), while each speaker had 7 different utterance for the same word. In order to evaluate the test (generalization) accuracy we have considered the most adverse circumstance where the training set comes from 5 speakers and the test set comes from the other 5 speakers. There are 315 words in both the training and the test set. Several different choices for computing the feature vectors (according to the NTA algorithm presented above) were considered as follows:

$$\mathbf{F1} = [\mathbf{f}_1],$$
$$\mathbf{F2} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3],$$
$$\mathbf{F3} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_2', \mathbf{f}_3'],$$
$$\mathbf{F4} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_1' \mathbf{f}_2', \mathbf{f}_3'],$$
$$\mathbf{F5} = [\mathbf{f}_1' \mathbf{f}_2', \mathbf{f}_3'].$$

The implementation of the SVM from (Junshui et al., 2003) was used, and the synthetic results are presented in Table 1. A Gaussian (RBF) kernel was used and the optimal value of its parameter γ is specified in the table. In addition the number „*sv*" of support vectors (giving an indication on the computational complexity of the SVM classifier) is also included in the table.

Table 1 Percentage of incorrectly classified words (PCIW) and its dependence on the choice of the feature vector F:

Feature vector:	**PICW (%)**	**SVs**	**Optimal γ**
$\mathbf{F1} = [\mathbf{f}_1]$	33.65	591	2
$\mathbf{F2} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3]$	22.22	668	0.55
$\mathbf{F3} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_2', \mathbf{f}_3']$	12.38	680	0.3
$\mathbf{F4} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_1' \mathbf{f}_2', \mathbf{f}_3']$	11.42	696	0.3

In further experiments, in addition to the above features, we considered the word length *L*. The results are summarized in Table 2:

Table 2: Percentage of incorrectly classified words (PCIW) and its dependence on the choice of the feature vector F (The word length was included in the feature vector):

Feature vector	**PICW (%)**	**SVs**	**Optimal γ**
$\mathbf{F1} = [\mathbf{f}_1, L]$	26.98	620	3
$\mathbf{F4} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_1' \mathbf{f}_2', \mathbf{f}_3', L]$	10.47 (best)	796	0.4
$\mathbf{F5} = [\mathbf{f}_1' \mathbf{f}_2', \mathbf{f}_3', L]$	13.01	991	0.9

It follows from the above tables that the choice of the components of the word feature vector F is of great importance for the recognizers' accuracy. A particularly significant feature is the word length *L*. The best recognition rate (10.47%) is obtained for $\mathbf{F4} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_1' \mathbf{f}_2', \mathbf{f}_3', L]$, while the computational complexity (in terms of arithmetic operations) of the SVM in this case is 70*796=55720. Note than $\mathbf{F}5 = \left[\mathbf{f}_1'\mathbf{f}_2', \mathbf{f}_3', L\right]$, with a total of 40 scalar features, does not significantly worsen the result (now PICW=13%) while the computational load reduces almost twice, to only 40*991=39640 operations. Note than for an average word length of L=60, the computational complexity of the NTA preprocessing scheme is typically less than 10% of the computational complexity of the SVM, therefore the SVM dominates within the recognition block formed of NTA + SVM. For comparison with standard approaches we employed a HMM model to do the classification task using the same dataset, resulting in a PCIW of 16.22% (i.e. worse than the 10.47% obtained with the NTA+SVM). In an attempt to further reduce the implementation complexity, several simpler neural networks were tried instead of the SVM. For instance a PCIW=24,76% is reached using an RBF neural network optimized for hardware implementation (Dogaru et al., 1996) which has a hardware complexity of M*11+79=99+79=178 much smaller than the complexity of the preprocessing scheme (in this case 79*60=4740). Using an Adaline (Haykin, 1999), with a computational complexity of only M*79=711, the performance drops to PCIW=28.6%.

6. CONCLUSION

A major problem in building compact, VLSI integrated speech classifiers based on isolated word recognition is the computational complexity of the classifier, usually implemented as an HMM model. This paper proposes a simple temporal averaging scheme, which generates a fixed size feature vector (with a dimension ranging from 40 to 80) associated with the temporal sequence representing an entire spoken word. Our method has the advantage of a dramatic reduction of the implementation complexity. As evaluated in Section 5, an average of 50.000 arithmetic computations per word were needed by our NTA+SVM algorithm to process a word, while for comparison 418.000 arithmetic operations per recognized word were reported in (Yoshizawa et al., 2006) for a VLSI-oriented recognizer using the HMM. Therefore our proposed recognition system has a lower complexity, of about 10 times smaller than of a HMM-based machine. Still knowing that the NTA requires only 10% of the recognizer complexity, there is a lot of potential in reducing the recognizer complexity further more, e.g. by using a simpler front-end than the Mel-cepstral one, and replacing the SVM algorithm with a more compact (and less computationally expensive) neural network. This is the aim of future research.

ACKNOWLEDGMENT

R. Dogaru acknowledges the research fellowship support of the Alexander von Humboldt Foundation and of the T.U. Darmstadt, Institute of Microelectronic Systems (Germany).

7. REFERENCES

Rabiner, L.R., (1989). “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”, *Proceedings of the IEEE*, **vol. 77**, no. 8, February, pp. 257-286.

Vapnik, V., (1999). “An overview of statistical learning theory”, *in IEEE transactions on Neural Networks* **Vol 10**, No 5, pp. 988-1000.

Binu K. Mathew, (2004). “The perception processor”, PhD-thesis, available from http://www.cs.utah.edu/~mbinu/perception/

Haykin, S., (1999). *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Company, New York.

Junshui Ma, Yi Zhao, and Stanley Ahalt, (2003). “OSU SVM Classifier Matlab Toolbox (ver 3.00)”, available from http://svm.sourceforge.net/docs/3.00/api/

Hearst, M.A., Scholkopf, B., Dumais, S., Osuna, E., & Platt, J. , (1998). “Trends and controversies - support vector machines”, *IEEE Intelligent Systems*, **Vol. 13** (4), pp. 18-28.

Audio-Visual Speech Processing Project, University of Carnegie Mellon, (2003). Advanced Multimedia Processing Lab, http://amp.ece.cmu.edu/projects/AudioVisualSpeechProcessing/download.htm

Dogaru, R., Murgan, A.T, Ortmann, S., Glesner, M., (1996). “A modified RBF neural network for efficient current-mode VLSI implementation”, in *Proceedings Micro-Neuro’96, (IEEE Press), Laussane 12-14 febr. 1996*, pp. 265-270.

Yoshizawa, S.; Wada, N.; Hayasaka, N.; Miyanaga, Y, (2006). “Scalable architecture for word HMM-based speech recognition and VLSI implementation in complete system”, *in IEEE Transactions on Circuits and Systems I*: **Volume 53**, Issue 1, Jan. 2006, pp. 70 – 77.