OPEN SET TEXT-INDEPENDENT SPEAKER RECOGNITION BASED ON SET-SCORE PATTERN CLASSIFICATION

Jiuqing Deng, Qixiu Hu

Department of Computer Science and Technology,
Tsinghua University, Beijing 100084, P. R. China
E-mail: djq97@mails.tsinghua.edu.cn, xxs-dau@mail.tsinghua.edu.cn

ABSTRACT

In this paper we propose a two-stage recognition schema for open set text-independent speaker recognition tasks. First we try to find a best matched model (which gets the best score) for the unknown speaker like many other systems. But then unlike other classical threshold selecting methods that make decisions based on the best score, we use the scores over a reference speakers set as a whole (called the set-score pattern): a binary classifier (e.g., an SVM) is then built to recognize acceptable and rejectable patterns. The results show that the set-score pattern classification method gives reasonably good performance. An obvious improvement has been seen compared to simple threshold selecting methods. And the painful procedure to choose a good threshold can be avoided too.

1. INTRODUCTION

Speaker recognition, one important branch of speech processing, is the process of automatically recognizing who is speaking by using speaker-specific information included in speech waves [1]. According to the constraints of the utterances, speaker recognition methods can be divided into two major categories: text-dependent methods and text-independent ones. The former require the speaker to provide utterances of words or sentences that the system prompts, and the latter do not. Speaker recognition can also be classified into "open set" and "close set" cases with respect to whether there is in the recognizing process a decision procedure to tell if "the unknown does not match any of the models" [2]. As for this paper, only open set text-independent speaker recognition is concerned.

Being a purely acoustic recognition task, text-independent speaker recognition faces several challenges, namely the high variability of channel properties and the question of choosing appropriate feature parameters and model structures to capture the unique characteristics of an individual voice. Currently, short-term cepstrum-based features are most commonly used. And among the various modeling techniques in application, the most successful are statistical models including hidden Markov models (HMM), vector quantization (VQ) and Gaussian Mixture models (GMM), etc [2].

In common cases, a speaker recognition system calculates the distance (sometimes in the form of a probability) between the unknown speaker and each known model and then chooses the best matched as the result. We call each distance a score. As for open set cases, a fixed or adaptive threshold is always used to determine if the best score is good enough so the result can be accepted. It is often difficult to find appropriate thresholds because the speakers and utterances vary so much. In this paper we propose a two-stage recognition schema to overcome this problem. First, we use a VQ-based classifier to find the best matched model for the unknown speaker: we called it the speaker classifier. Second, a binary classifier (an SVM) is applied and rejection decisions are made based on the scores over a reference speakers set (called the set-score pattern), so that a threshold selecting procedure can be avoided: we called it the set-score pattern classifier.

This paper is organized as follows: first we discuss the structure of the speaker classifier in section 2, and then the set-score pattern classifier in section 3. Finally, we present experimental results in section 4, followed by the conclusions in section 5.

2. THE SPEAKER CLASSIFIER

The most widely used statistical modeling techniques in text-independent speaker recognition are VQ and GMMs. Considering its simplicities and robustness, we choose VQ as the basic method in the first stage.

In a VQ-based recognition system, a speaker is modeled as a set of feature vectors generated from his/her voice samples. When training, the speaker models are constructed by clustering the feature vectors into N separate clusters called cells. Each cell is then represented by a code vector, which is often the average vector of that

cell. The resulting set of code vectors is called a codebook, and is stored in the speaker database. In the recognizing stage, an input utterance is vector-quantized by the codebook of each reference speaker. The VQ distortion in the codebook is accumulated (over the entire utterance) and is used for making the result determination (Fig. 1) [6].

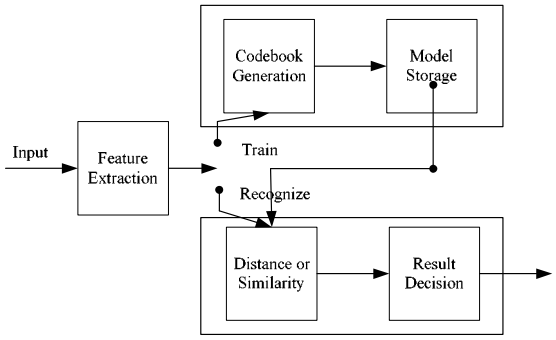

Fig. 1: Block diagram of a VQ-Based recognition system

In a system with M reference speakers, denote the codebook for speaker i as: $C_i = \{c_{i1}, c_{i2}, \cdots, c_{iN}\}$, $i = 1, \cdots, M$, where N is the size of a codebook and c_{ij} is the code vector of codebook i's jth cell. The matching of an unknown speaker to the reference speakers set is then performed as follow: the sequence of feature vectors $X = \{x_1, x_2, \cdots, x_T\}$ is extracted from the utterance, the distance between X and every C_i : $d(X, C_i)$, is calculated by some distortion measurement, and the codebook with minimal distance is choosed as the recognition result, i.e.:

$$result = \arg\min_i \{d(X, C_i)\} \qquad (2.1)$$

An simple algorithm to calculate the distance $d(X, C_i)$ is to accumulate the distance (to be simplified, we use Euclidean distance) between every feature vector x_t and the corresponding nearest code word $c_{ik(t)}$ in codebook C_i:

$$d(X, C_i) = \sum_t d(x_t, c_{ik(t)}) \qquad (2.2)$$

Various kinds of modification to the distance measurement, for example, weighted Euclidean distance, have been made to the VQ-based systems and have achieved more or less performance improvements.

With the basic system mentioned above, we can now define the terms of "score" and "set-score" in detail. For a feature vector sequence X and a codebook C_i, $d(X, C_i)$ is called a score of X on codebook C_i. For scores of a feature sequence X and a reference set S with M speakers (represented by M codebooks), vector $d_S = \{d(X, C_1), d(X, C_2), \cdots, d(X, C_M)\}$ is called a set-score of X on speakers set S. The speaker classification process can be viewed as the procedure of searching for the best score of the feature vector sequence (derived from the utterance) on the codebooks then. Here we can see that the set-score on the speakers set S is a byproduct.

3. THE SET-SCORE PATTERN CLASSIFIER

3.1. Rejection considerations for open set recognition

For close set systems, when the best score is found, the corresponding model is output as the recognition result. But when dealing with open set cases, one more question must be answered: is the score good enough or is it only better than others (the match might be quite bad in fact).

A common way is to define a threshold for each model: if the score is better than the threshold we accept the result, otherwise reject it. But it is not easily applicable because speakers and utterances vary all the time and it requires quite some specialized knowledge and empirical information to get a good threshold.

Further more, we can see that the threshold selecting methods use only part of the score information (the best or the N-best). How if we use the entire scores information? Here comes the idea. We don't care too much on the numerical value of a single score or several, on the other hand, we regard the scores on the speakers set (the set-score) as a pattern of the speaker and try to apply a classifier on them. Obviously it is a binary classification problem: to be or to be not. Several methods have been considered, such as Multi-Layer Perceptron (MLP) and SVMs. We decide to use SVMs because they have strong generalization capacities and are much faster (using SMO Sequential Minimal Optimization algorithm [5]).

3.2. Support Vector Machines

In recent years, Support Vector Machines have been used for a wide variety of classification problems. They have shown good generalization capacities in application. In short, SVMs map the input feature vectors (the input space) into a high dimensional space (called the feature space), and then try to find an optimal hyper-plane (a linear classifier: $f(x) = x \cdot w + b$) to separate the train samples. The non-linear mapping provides SVMs the power to accomplish complex classification tasks, and the simple architecture of a linear classifier in the feature space supplies them a good control of generalization capacities.

An SVM works as follow: consider the training sample vectors of two classes,

$$(x_i, y_i),\ x_i \in R^n,\ y_i \in \{+1, -1\}.$$

For linear separable cases, the SVM try to find a hyper-plane in all possible separating hyper-planes which has the

maximal margin. We can do that by minimizing $\|w\|^2$, subject to the constraints:

$$y_i(x_i \cdot w + b) \geq 1, \ \forall i \qquad (3\text{-}1)$$

The problem can be converted to a convex quadratic programming problem and in the problem the training data appears only in the form of dot products, $x_i \cdot x_j$ [3].

As for non-linear cases, we first map the data to a Euclidean Space H using mapping $\Phi : R^n \rightarrow H$, then search for the optimal hyper-plane in space H instead of R^n. Like above, the training algorithm depends on the data only through dot products in H (i.e., $\Phi(x_i) \cdot \Phi(x_j)$). So, if we can find a kernel function K: $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$, we can even avoid the explicit form of mapping Φ. Practically we are more interested in the kernel function than the mapping, as *Mercer's Condition* can tell what kind of kernel functions have corresponding mapping pairs $\{H, \Phi\}$ [3].

There are various kinds of kernel functions used in practice. Two kernels used in our systems: the polynomial kernel (Eq.3-2) and the Gaussian radial basis function (RBF) kernel (Eq.3-3).

$$K(x, y) = (x \cdot y + 1)^p \qquad (3\text{-}2)$$

$$K(x, y) = e^{-\|x-y\|^2/2\sigma^2} \qquad (3\text{-}3)$$

One more thing to be mentioned is the *Soft Margin*, which is introduced to handle non-separable cases. Constraints (3-1) are relaxed by introducing positive slack variables ξ_i:

$$y_i(x_i \cdot w + b) \geq 1 - \xi_i, \ \xi_i \geq 0, \ \forall i \ . \qquad (3\text{-}4)$$

And now the optimizing problem is to minimize $\|w\|^2 + C\sum_i \xi_i$ [4]. Here C is the penalty function and controls the complexity/generalization capacities of the machine. Roughly speaking, when C becomes larger, the machine fits closer to the training data but lose some generalization capacities, or otherwise.

3.3. Set-score pattern classification

Denote S_0 for the set of speakers which the system "knows", which is set up by the speaker classifier mentioned in Sec. 2: a rejection decision: if the speaker belongs to S_0 or not, is to be made. Denote S_r for the reference speakers set on which the set-score is calculated. S_r is set up in similar ways as those of S_0. S_r is not necessarily equal to S_0. In fact, we define 2 operation modes with respect to the characteristics of S_r: if S_r is equal to S_0, we call that the system is working in "self-set" mode, otherwise, "reference-set" mode. In "self-set" mode, the codebooks of S_r are the same as those of S_0, thus do not need to be trained separately. In "reference-set" mode, if S_r is not a subset of S_0, extra training is required.

After the codebooks of S_r have been constructed, the speaker's speech is divided into subsequences of equal lengths and each subsequence's set-score d_S on S_r is calculated. A classification machine (in our case, an SVM) is then trained using the set-scores: for each speaker *i* in S_0, a optimal classification hyper-plane between set-scores of *i* and those of others is built.

When recognizing, the speaker classifier first find out the best matched speaker for the input speech data, then set-scores of the input is calculated and then classified by the speaker's classification hyper-plane.

4. EXPERIMENTS

A database of 200 speakers (120 males and 80 females) collected in laboratory is used in our experiments. The wave data were recorded at the sampling rate of 8.0 kHz and the quality of 16 bits per sample. The average duration of the training samples is about 2 minute per speaker. Another 60s speech sequence for each speaker is recorded for testing purpose.

The feature extraction process is performed using the following steps:

· Divided into 60 ms frames, shifted by 30 ms.

· DC value removal, high-emphasis filtering with filter $1/(1 - 0.95z^{-1})$, hamming windowing.

·16 LPC-cepstral coefficients are calculated and then concatenated with the derivatives of themselves. Thus the dimension of the feature vector is therefore 16 + 15 = 31.

4.1. The baseline system (the speaker classifier)

The baseline system is VQ-based with the codebook size of 64. All the 200 speakers are included in this system. For each speaker, the first half of the training speech (1 minute) is used to train his/her codebook. When recognizing, the test speech is divided into three different subsequences of the lengths 12 seconds, 6 seconds and 3 seconds. The recognition results are shown below:

Systems and parameters	Recognition rate
VQ, 3.0s, codebook size 64	88%
VQ, 6.0s, codebook size 64	97.5%
VQ, 12s, codebook size 64	98.5%
VQ, 6.0s, codebook size 80	98.5%

Tab. 1: Recognition results of the baseline systems

We can conclude that the system can achieve quite high accuracy when the testing speech is no shorter than 6

seconds. Thus out comes our speaker classifier: a 64-code VQ system with 1 minute training speech and 6 seconds testing speech.

4.2. The set-score pattern classifier

First we test the "self-set" mode recognition. We choose 150 speakers randomly as the known speakers set S_0, and the others as the unknown. The codebooks are simply copied from the baseline system. Then each speaker's wave data is divided into 6 second sequences to train the set-score classifier (if not equal, a compensatory factor is applied). We try different kinds of kernels and parameters when training the SVMs. With the trained system, we divided the 1 minute testing speech into 6s sequences (9 sequences per speaker) and test each. The results are listed in Tab. 2:

Methods and parameters		Accept (all/bd)	Reject (all/bd)
VQ (adaptive thresholding)		88%	90%
VQ/SVM (self set)	P/D3/C100	71%/74%	93%/87%
	P/D5/C100	74%/76%	88%/81%
	R/C100	76%/79%	94%/86%
	R/C100/1m	84%/79%	94%/93%
	R/C50/2m	89%86%	96%/92%
	R/C200/2m	89%/86%	96%/93%

Tab. 2: Recognition correct rate of VQ/SVM systems

NOTE: The adaptive threshold selecting method has not been fully optimized. Acronyms: P: a polynomial kernel, R: an RBF kernel, D (polynomial kernels only): degree, C: the penalty function, 1m means using another half of the 2min speech when training the SVM (the default is to reuse the first half 1min), 2m means using the entire 2min speech, "all": all the scores are used, "bd": the best discarded.

To test the efficiency of the set-score pattern, we attempt to discard the best score both in the training and the recognizing stage, performance degrades, but not too much (see Tab.2). We can also see that when training data increase, the correct rate increases quite much too: this is mainly because 1 min speech can only produce 10 set-scores, which may be too few for an SVM classifier. RBF kernels also have given better results in our experiments.

We also test the "reference set" mode. The 200 speakers is divided into 3 sets: 100 for the reference set, 50 for the known, and 50 for the unknown. The recognition rate (R/C100) for the reference and the known set are 76% and 68%, and the rejection rate for the unknown is 93%, not so good yet. One reason is that the reference set is smaller and less representative.

A typical error rate distribution of speakers for a good result (R/C200/2m) is show in Fig. 2. We can find out that for most speakers, the error rates are low: for acceptable cases, more than 50% speakers (see 0% error rate column) pass all 9 tests, more than 85% speakers fail no more than 1 test), and the numbers for rejections cases are even better (75% and 93%, respectively).

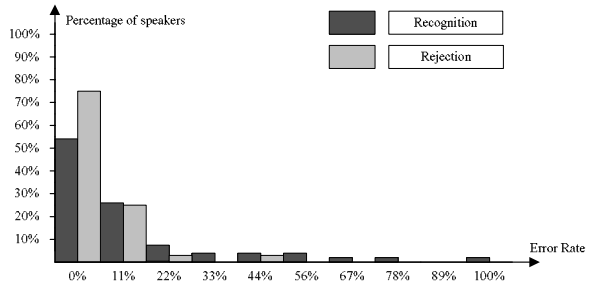

Fig. 2: Error distribution of speakers

5. CONCLUSIONS

There are two main points in our experiments considerations. The first is to use as much information as possible: i.e., we use a set-score other than a single best score for rejection decision. The second is to build a simple system with a better generalization capacity and then try to make improvements.

From the results we can conclude that both set-score pattern and SVMs are quite robust. And with enough training data, they are also efficient. Improvements can also be made too: better score representation, other classification methods can be tried, etc.

6. REFERENCES

[1] G. Doddington, "Speaker Recognition - Identifying People by their Voices," *Proc. of the IEEE*, 73(11), pp.1651-1664, 1985.

[2] S. Furui, "Recent Advances in Speaker Recognition," *First International Conference, Proc. of AVBPA*, pp.238-250, 1997.

[3] V. Vapnik, "The Nature of Statistical Learning Theory," *Springer-Verlag*, New-York, 1995. (Chinese version: *Tsinghua University Press*, Beijing, 2000).

[4] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Knowledge Discovery and Data Mining*, 2(2), 1998.

[5] J. Platt, "Fast Training of Support Vector Machines using Sequential Minimal Optimization," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, eds., MIT Press, 1998.

[6] Qin Jin, Luo Si, and Qixiu Hu, "A high performance text-independent speaker identification system based on BCDM," *Proc. of ICSLP*, vol. 2, pp. 133-136 , 1998.