Vector-Based Spoken Language Recognition using Output Coding

Haizhou Li, Bin Ma, Rong Tong

Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
{hli, mabin, tongrong}@i2r.a-star.edu.sg

10.21437/Interspeech.2006-139

ABSTRACT

The vector-based spoken language recognition approach converts a spoken utterance into a high dimensional vector, also known as a *bag-of-sounds* vector, that consists of *n*-gram statistics of acoustic units. Dimensionality reduction would better prepare the *bag-of-sounds* vectors for classifier design. We propose projecting the *bag-of-sounds* vectors onto a low dimensional SVM output coding space, where each dimension represents a decision hyperplane between a pair of spoken languages. We also compare the performances of the output coding approach and the traditional low ranking approximation approach using latent semantic indexing (LSI) on the NIST 1996, 2003 and 2005 Language Recognition Evaluation (LRE) databases. The experiments show that the output coding approach consistently outperforms LSI with competitive results.

Index Terms: spoken language recognition, vector space model, multiclass classification

1. INTRODUCTION

Vector-based spoken language recognition (LID) [1] has emerged recently as an effective phonotactic approach. It is different from the parallel PRLM method [2] in that it represents a spoken utterance as a vector. In this approach, a spoken document is represented by a *bag-of-sounds* vector, similar to the document vector of a text document. A *bag-of-sounds* vector is expanded in a high dimensional space by the statistics of the acoustic units, such as the phone *n*-gram. In this way, the vector-based approach translates the LID task into a typical vector classification problem in pattern recognition. Many advanced algorithms for vector classification can then be readily applied. Therefore, one of the challenges is to handle the high dimensional vectors effectively. Latent semantic indexing (LSI), which uses the truncated singular value decomposition (SVD) as a low-rank approximation, has been studied for *bag-of-sounds* dimensionality reduction [1]. Another established algorithm is linear discriminant analysis (LDA). It combines the features of the original data so as to reduce the dimensionality of original data in a way that most effectively discriminates classes [3][4].

In LDA, the problem of finding a linear discriminant function is formulated as the problem of minimizing a criterion function. The common criterion function for classification purposes is the classification error, the average loss incurred in classifying the set of training samples. The minimum squared-error (MSE), is one of the solutions to the problem, by solving a set of linear equations. The procedure involves all the training samples and assumes that within-cluster scatter matrix is nonsingular. Recently, support vector machines (SVM) have attracted much attention as an alternative solution. First, in MSE type of solutions, we are over concerned with the distributions of all the samples. In contrast, a SVM system places a hyperplane in a high dimensional space so that the hyperplane has maximum margin. Therefore, SVM is more interested in the decision boundaries than the sample distributions themselves. Second, SVM relies on preprocessing of the samples to represent patterns in dimensions, typically much higher than the original feature space. The idea is that, with an appropriate nonlinear mapping to a sufficiently high dimension, samples from two classes can always be separated by a hyperplane. SVM has shown to be an effective machine learning method for pattern classification in high dimensional spaces. Many attempts, for instance, in text categorization [5] have been successful.

There are two typical tasks in LID, language recognition and language verification. The former is formulated as a closed-set identification problem assuming that the test languages are known to the system. The latter is formulated as an open-set problem where the test languages can be new to the system. In the former case, a vector-based, multiclass SVM system might serve the purpose. However, in the latter, it is desirable to reduce the *bag-of-sounds* vector to a lower dimensional space so that the traditional hypothesis test theory can be practically applied. In this paper, we will study a new approach to dimensionality reduction by using SVM with language verification as the application. This paper is organized as follows. In Section 2, we briefly discuss vector-based LID technique. In Section 3, we propose using SVM for dimensionality reduction. In Section 4, we report experiments on the NIST LRE databases. Finally we conclude in Section 5.

2. VECTOR SPACE MODELING

Suppose that we have an inventory of $V = \{V_1, V_2, ...V_F\}$ universal acoustic units. As discussed in [6], an acoustic unit can be either a linguistically defined phone or an acoustically defined unit. For each sound sequence generated from the universal acoustic tokenizer, we derive a spoken document vector from the *n*-gram of the acoustic units. Suppose that we have V acoustic units in F parallel phone recognizers, also known as PPR (see Figure 1), each producing a sound sequence. We can derive a large composite document vector by concatenating F vectors resulting from the individual phone recognizers, for instance, resulting in a *bag-of-sounds* vector of $V_1 \times V_1 + ... + V_F \times V_F$ dimensions in the case of bigrams. In this way, the utterances for each

language are represented as a collection of vectors. The LID task can be seen as a multiclass problem in the *bag-of-sounds* vector space. Details of *bag-of-sounds* method can be found in [1].

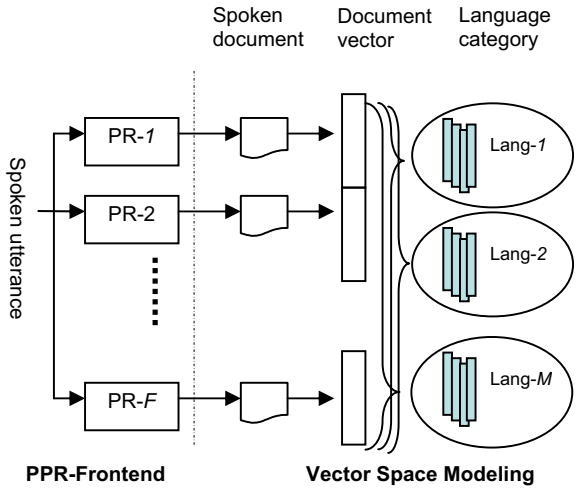

Figure 1: LID systems with parallel phone recognition frontend and vector space modeling backend

3. CONTINUOUS OUTPUT CODING

A *bag-of-sounds* vector typically has a dimensionality of tens of thousands given a reasonable size of acoustic inventory V. To effectively apply pattern classification techniques, it is desirable to reduce this dimensionality to less than a few hundreds. Dimensionality reduction is typically achieved through the linear combinations of samples in their original space. Linear combinations are particularly attractive because they are simple to compute and analytically tractable.

3.1 Latent semantic indexing

A technique that significantly reduces the feature dimension is the use of SVD. A collection of *bag-of-sounds* vectors can be arranged in the form of a $K \times D$ term-document matrix, *H*. We can decompose the $K \times D$ term-document matrix into the product of three matrices: $H = USV^T$, where U is a $K \times R$ left singular matrix with rows u_k, where $1 \le k \le K$, S is a $R \times R$ diagonal matrix of singular values, with $s_1 \ge s_2 \ge ... \ge s_R > 0$; and V is a $D \times R$ right singular matrix with rows v_d, where $1 \le d \le D$. Both the left and the right singular matrices, *U* and *V*, are column-orthonormal.

If we retain only the top *Q* singular values in matrix *S* and zero out the other (*R*-*Q*) components, the feature dimension can be effectively reduced to Q, which is often much smaller than *R*. By doing so, the three matrices are much smaller in size than those in USV^T, which greatly reduces the computational requirements. We can therefore compare spoken documents in this new Q dimensional space. Any document or query represented by a vector in the original *K*-dimensional space can now be transformed into a *Q*-dimensional vector. These reduced vectors are then used to train all language classifiers and perform LID. Given a test spoken document $\mathbf{x}$, using the left singular matrix *U*, we can construct a document vector $\mathbf{y}$ in the *Q*-dimensional space,

$$\mathbf{y} = \left(US^{-1}\right)^t \mathbf{x} = \mathbf{W}_{LSI}^t \mathbf{x} \tag{1}$$

This approach is known as latent semantic indexing (LSI), which is based on the assumption that there is some underlying latent semantic structure in the term-document matrix that is corrupted by the wide variety of terms used in the documents. The LSI dimensionality reduction allows us to retain semantic structure in the low dimensional space. However, there is no reason to assume that these components must be useful for discriminating different classes.

3.2 Continuous output coding

In LDA, we project a vector $\mathbf{x}$ from a *K*-dimensional space to a *C*-dimensional space by *C* discriminant functions [4] in the directions that are efficient for class discrimination.

$$y_i = \mathbf{w}_i^t \mathbf{x} \qquad i = 1,...,C \tag{2}$$

If we consider y_i as components of a vector $\mathbf{y}$ and the weight vectors $\mathbf{w}_i$ as the columns of a $K \times C$ matrix $\mathbf{W}_{LDA}$, then the projection can be written as a single matrix equation

$$\mathbf{y} = \mathbf{W}_{LDA}^t \mathbf{x} \tag{3}$$

$\mathbf{W}_{LDA}$ is similar to $\mathbf{W}_{LSI}$ except that it is achieved by solving *C Fisher linear discriminants* $J(\mathbf{w}_i)$ [4]. As a variant of LDA, SVM has been shown to be effective in separating high dimensional vectors in 2-class problems, in which SVM effectively projects the high dimensional vector $\mathbf{x}$ into a scalar value $y = f(\mathbf{x})$; Suppose that we have a collection of *L* support vectors, a SVM is constructed from the sums of a kernel function $k(\cdot,\cdot)$.

$$f(\mathbf{x}) = \sum_{l=1}^{L} \alpha_l t_l k(\mathbf{x}, \mathbf{x}_l) + d \tag{4}$$

where t_l are the ideal outputs, $\sum_{l=1}^{L} \alpha_l t_l = 0$ and $\alpha_l > 0$. For a linear kernel $k(\mathbf{x}, \mathbf{x}_l) = \mathbf{x}^t \mathbf{x}_l$, Eq.(4) can be easily rewritten as

$$\begin{aligned} y = f(\mathbf{x}) &= \sum_{l=1}^{L} \alpha_l t_l \mathbf{x}_l^t \mathbf{x} + d \\ &= \left(\sum_{l=1}^{L} \alpha_l t_l \mathbf{x}_l + \mathbf{d} \right)^t \mathbf{x} = \mathbf{w}_c^t \mathbf{x} \end{aligned} \tag{5}$$

where $\mathbf{d} = [d, 0, ..., 0]$. Interestingly, once we train a SVM, we can collapse all the support vectors $\mathbf{x}_l$ into a single weight vector $\mathbf{w}_c = \sum_{l=1}^{L} \alpha_l t_l \mathbf{x}_l + \mathbf{d}$ similar to the weight vectors $\mathbf{w}_i$ in Eq.(2). If we see $\mathbf{w}_c$ as the columns of a $K \times C$ matrix $\mathbf{W}_{SVM}$, then the projection can be written as

$$\mathbf{y} = \mathbf{W}_{SVM}^t \mathbf{x} \tag{6}$$

$\mathbf{W}_{SVM}$ is similar to $\mathbf{W}_{LDA}$ or $\mathbf{W}_{LSI}$ except that it is constructed by *C* SVM decision hyperplanes. Each SVM is represented by a weight vector $\mathbf{w}_c$. In this way, the *C*

SVM outputs form a reduced dimensional space, which is also known as output coding [7].

Output coding, also known as error-correcting output coding, is a general method for solving multiclass problems by reducing them to multiple binary classification problems [7]. Typically, output codes are defined as discrete codes of 0 and 1. Using SVM output as the output coding bit *b*, we have $b=1$ if $f(\mathbf{x})>0$, and $b=0$ otherwise. Let's describe the technique of output coding through a simple example: the task of classifying a vector into the $M=4$ classes. Each language is assigned with a *C*-bit vector with $C>\log_2 M$, for example, Chinese - 0110110001, English - 0001111100, French -1010101101 and Korean - 1000011010.

A bit in the bit-vector represents the output of an individual SVM. The bit-vector is an output collection of *N* individual SVMs. This is also known as an ensemble classifier, which makes collective decision based on a number of individual decisions. Using output coding, a class is encoded by a centroid code. The ensemble classifier is able to correct some errors that individual classifiers make, thus also known as error-correcting output coding. The greater the distance between the centroid codes, the better discriminability that a multiclass classifier has. In general, larger code size leads to better performance. Some recent work improves the performance of output coding by relaxing the output codes from discrete coding to continuous coding [7]. In practice, Eq.(6) is implemented like this. First, we train a set of *C* independent SVM classifiers $\{f^1(\cdot), f^2(\cdot),..., f^C(\cdot)\}$, then we represent each of training or test utterance $\mathbf{x}$ as a vector of *C* real-valued SVM outputs $\mathbf{y}=\{f^1(\mathbf{x}), f^2(\mathbf{x}),..., f^C(\mathbf{x})\}$, known as continuous output coding (COC).

3.3 Designing of output codes

Now the question is how to construct the *C* SVMs. A natural way is to construct *C* independent SVMs, each deciding a target language vs. the rest. In doing so, we denote the samples labeled with the target category as the positive samples, and the rest as the negative samples. The code size of the so defined output code vector equals the number of classes *M.* In discrete coding, for an ensemble of binary classifiers, truly meaningful values of code size *C* lie in the range $[\log_2 M, (2^M-1)/2]$. A code size of $C<\log_2 M$ cannot even assign a distinct bit-vector to each label. At the other extreme, a code of size $C>(2^M-1)/2$ must contain duplicate columns, which means two individual classifiers learning the same task. Studies show that a code size of $C\geq 10\log_2 M$ gives reasonably good performance, while increasing code size improves performance at the cost of higher computational need [7]. Note that, with the same code size, COC consistently outperforms discrete coding in multiclass classification tasks. We adopt COC by having pairwise SVM classifiers as the individual classifiers. Suppose that we have *M* language categories, we build $M\times(M-1)/2$ classifiers, with each discriminating a language pair, resulting in a COC vector of $C=M\times(M-1)/2$ dimensions.

To understand why one might expect our COC coding to work, consider the problem of learning to classify three languages. Each language pair has its distinctive distinguishing characteristics as follows:

	Tonal	Syllabic
Chinese	yes	yes
Japanese	no	yes
English	no	no

If we build $C=3\times(3-1)/2=3$ SVMs, each discriminating a language pair, one can expect the Chinese-Japanese classifier to learn the strong association between tonal characteristics and Chinese, in order to differentiate it from Japanese. Likewise, the Japanese-English classifier will learn the strong association between a syllabic structure and Japanese. During the training of a Chinese-Japanese SVM classifier, we take Chinese data as the positive samples and Japanese data as the negative samples, leaving the English data out. The same principle applies to the training of all SVMs. To the Chinese-Japanese SVM, when a Chinese *bag-of-sounds* vector is presented, the SVM is trained to derive a positive value $f(\mathbf{x})>0$. In contrast, the SVM will respond to a Japanese vector with a negative value $f(\mathbf{x})<0$. The SVM output value for an English vector is to be decided by the SVM depending on how the SVM sees English from the point of view of the Chinese-Japanese decision hyperplane.

4. EXPERIMENTS

We follow the experiment setup in the NIST LRE tasks[1]. In the 1996 and 2003 tasks, 12 known languages are tested, with Russian being the out-of-language (OOL) in the 2003 test. In the 2005 task, 8 languages, a subset of the 1996 and 2003 languages, are tested, with German being the OOL.

Training sets came from three corpora [6], namely: (i) the 3-language IIR-LID database; (ii) the 6-language OGI-TS (Multilanguage Telephone Speech) database; and (iii) the 12-language LDC CallFriend database. Both IIR-LID and OGI-TS are telephone speech with phonetic transcriptions. They are used for acoustic modeling. In addition, the CallFriend database was used for constructing *bag-of-sounds* vectors and designing classifiers [6]. It contains telephone conversations of the same 12 languages as are in the 1996 and 2003 NIST LRE tasks, with 3 languages having 2 accented versions. As a result, we have $M=15$ (see Figure 1). The three databases are independent of each other.

In NIST LRE tasks, there are 3 different duration settings, 3, 10, and 30 seconds. The 1996 evaluation data consist of 1,503, 1,501 and 1,492 sessions of 3, 10, and 30 seconds, respectively. The 2003 and 2005 evaluation data consists of 1,280 and 3,662 sessions per duration. In classifier design, each conversation in the CallFriend database is segmented into overlapping sessions, resulting in about 12,000 sessions for each of 3,10, and 30 seconds duration per language.

4.1 LSI vs COC

We begin by comparing two dimensionality reduction approaches, namely LSI and COC in the language

[1] http://www.nist.gov/speech/tests/index.htm

verification, where each utterance is evaluated against a hypothesized language to produce a confidence score. We report equal error rates (EER%) on evaluation data. The confidence score can then be compared with a threshold, if needed, for decision making. The *yes* decision accepts the hypothesized language as the detected language, while the *no* decision rejects it.

Given an acoustic inventory $V = \{48,39,52,51,32,36\}$ [6] with $F = 6$ parallel phone recognizers, we derive a *bag-of-sounds* vector of 11,708 dimensions [6] for each input utterance. After dimensionality reduction, we construct Gaussian mixture models (GMM) for each target language m^+ and their competing languages m^-. As such, for each target language, we build a pair of GMMs $\{m^+, m^-\}$. For language verification, we need to evaluate the probability of a hypothesized language model m^+ for a given test utterance O, $P(m^+ \mid O)$. However, the output of a GMM system gives $P(O \mid m^+)$. By making the assumption that all languages are equiprobable, we approximate the *posterior* probability $P(m^+ \mid O)$ by Bayes' theorem:

$$\log P(m^+ \mid O) = \log P(O \mid m^+) - \log P(O \mid m^-) \qquad (7)$$

Eq.(6) gives a relative log-likelihood score between the target language and its competing languages. Intuitively, $\log P(m^+ \mid O)$ reflects how the target model overtakes the competing models with respect to the input utterance, thus serving as the confidence score of a test utterance O being hypothesized as m^+. A series of systematic experiments on the GMM size suggest us 512 mixtures for m^- and 64 mixtures for m^+ because m^+ has much more training data than m^- has in our experiments.

For a system of $M = 15$ target languages, the COC reduces the 11,708-dimension *bag-of-sounds* vectors to $C = 105 = 15 \times (15-1)/2$ dimensional vectors. To establish fair comparison, we also apply LSI to reduce *bag-of-sounds* vectors to 50, 105 and 200 dimensions, from which we further train GMM models $\{m^+, m^-\}$ for each language. Without loss of generality, we only test 30-sec samples in this experiment as reported in Table 1. Note that COC consistently outperforms LSI approach across all the settings. This can be explained by the fact that COC describes the "difference" between the languages while LSI describes the "distribution" of data in general.

4.2 Overall performance comparison

Language recognition technology has gone through many years of evolution. Many results have been published in the literature [8] on the NIST 1996, 2003 and 2005 LRE tasks, which provides good benchmarks for new technology development. We further validate our COC approach by carrying out comprehensive benchmarking across different durations, namely, 3, 10 and 30 seconds as reported in Table 2, which are among the best reported results on the NIST databases [8]. COC consistently outperforms PPRLM approach as reported in [6]. The COC vector-based subsystem has contributed to the final submission to NIST 2005 LRE representing IIR site.

Table 1. EER% for NIST 96/03/05 30-sec tests

System	1996	2003	2005
LSI Q=50	4.29	4.84	7.47
LSI Q=105	3.63	4.83	7.35
LSI Q=200	3.09	4.36	6.58
COC C=105	2.75	4.02	5.78

Table 2. EER% for NIST 96/03/05 using COC

System	30-second	10-second	3-second
1996	2.75	8.23	21.16
2003	4.02	10.97	21.66
2005	5.78	12.48	24.23

5. CONCLUSIONS

We have proposed a COC approach for dimensional reduction. The experiments have shown that COC is an effective dimensionality reduction approach. It consistently outperforms LSI in language verification tests. The main contributions of this paper are in two areas: (i) we have formulated COC, that was originally used for multiclass pattern classification, for pattern verification; and (ii) we have applied the COC to a *bag-of-sounds* LID framework and achieving one of the best reported results. Without loss of generality, we have presented using pairwise SVM outputs as the output coding scheme. We believe that there exist many other designing schemes of output coding in the proposed framework that we will explore in the future.

6. REFERENCES

[1] Li, H. and Ma, B., "A phonotactic language model for spoken language identification," *Proc. ACL*, 2005.

[2] Zissman, M. A., "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Trans. Speech and Audio Processing*, Vol. 4, No. 1, pp. 31-44, 1996.

[3] Kim, H., Howland, P. and Park, H., "Dimension reduction in text classification with support vector machines", *Journal of Machine Learning Research*, 6, 2005, pp.37-53.

[4] Duda, R. O., Hart, P. E., Stork, D. G., *Pattern Classification*, John-Wiley & Sons, 2001.

[5] Joachims, T., "Text categorization with support vector machines: Learning with many relevant features", *Proc of European Conference on Machine Learning*, Berlin, 1998, pp.137-142.

[6] Li, H., Ma, B. and Lee, C.-H., "A Vector Space Modeling Approach to Spoken Language Identification", to appear in *IEEE Trans. on Audio, Speech and Language Processing*, Vol. 15, 2007.

[7] Crammer, K., and Singer, Y., "Improved Output Coding for Classification Using Continuous Relaxation," *Proc. NIPS*, 2000.

[8] Gauvain,J.L.,Messaoudi, A., Schwenk, H., "Language recognition using phone lattices", *Proc. ICSLP*, 2004.