Joint Estimation of Thresholds in a Bi-threshold Verification Problem

Simon Ho and Brian Mak

Hong Kong University of Science & Technology
Department of Computer Science
Clear Water Bay, Hong Kong
{csho, mak}@cs.ust.hk

Abstract

Verification problems are usually posted as a 2-class problem and the objective is to verify if an observation belongs to a class, say, A or its complement A'. However, we find that in a computer-assisted language learning application, because of the relatively low reliability of phoneme verification — with an equal-error-rate of more than 30% — a system built on conventional phoneme verification algorithm needs to be improved. In this paper, we propose to cast the problem as a 3-class verification problem with the addition of an "in-between" class besides A and A'. As a result, there are two thresholds to be designed in such a system. Although one may determine the two thresholds independently, better performance can be obtained by a joint estimation of these thresholds by allowing small deviation from the specified false acceptance and false rejection rates. This paper describes a cost-based approach to do that. Furthermore, issues such as per-phoneme thresholds vs. phoneme-class thresholds, and the use of bagging technique to improve the stability of thresholds are investigated. Experimental results on a kids' corpus show that cost-based thresholds and bagging improve verification performance.

1. Introduction

Verification technology is widely used in areas like security systems, biometrics, and learning systems. Examples in the speech domain include speaker verification as a biometric identification method, language verification and accent verification for many telephony automated systems, as well as phoneme verification for language learning. In most systems, verification is posted as a hypothesis testing problem. It is a 2-class problem in which the null hypothesis is that an observation belongs to a class, say, A and the alternative hypothesis is that it does not — or, in other words, the observation comes from the class, A' (A's complement). A common metric to evaluate a verification system is its *equal error rate* (EER) which is obtained from the threshold that gives equal *false acceptance rate* (FA) and *false rejection rate* (FR).

We are interested in the use of verification technology in a *computer-assisted language learning* (CALL) application. In a CALL system like [2], a user is asked to produce several utterances for evaluation and an average score is reported. In [1], evaluation scores are reported at both sentence level and speaker level. One problem with [2, 1] is that the feedback in terms of an overall score is not too useful for learning: the speaker still does not know how to improve his pronunciation based on the score since he does not know what is wrong with his pronunciation. Witt and Young [5] compute confidence scores for each phoneme in an utterance in a CALL system. The significance of the work is that one may now pinpoint the pronunciation accuracy at the phoneme level, which we believe is necessary. Leung and Siu [4] further propose the combination of confidence measures computed from both MFCCs features and articulatory features, and report better phoneme verification performance. Although these works represent great progress in the development of CALL applications, a major difficulty we encounter is the relatively low reliability of phoneme verification. Table 1 shows the EERs we find in a system using conventional verification method. These figures are simply too high to build a useful CALL application.

Table 1: Global and Average EER

EER using phone-independent threshold	38.03%
Average EER using phone-dependent thresholds	31.66%

In this paper, we propose to re-cast the verification problem as a 3-class verification problem with the addition of a new "in-between" class besides A and A'. The main idea is that there are often pronunciations which are similar but do not reach the exact acoustic targets; classifying them as incorrect may not be the best approach. This is especially true for a beginner who is also a non-native speaker with a foreign accent. Or, there are times when the system is not confident about its verification result, then it will be better to be more conservative than to give a wrong "reinforcement feedback" to the learner. We believe it is more reasonable to create an "in-between"

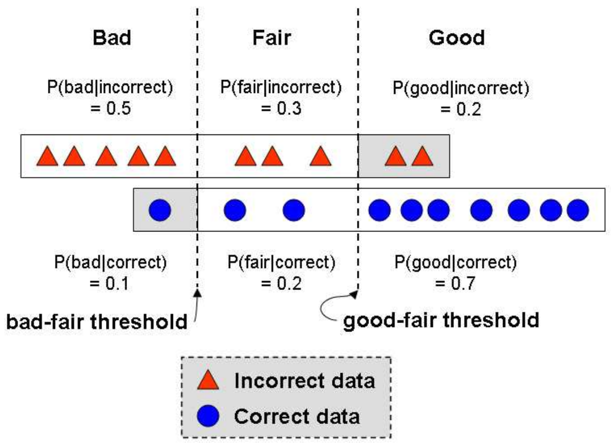

Figure 1: Definition of the two thresholds in our bi-threshold verification problem.

class to capture these inexact but reasonable pronunciations; and, a pronunciation falling to this "in-between" class will not be considered wrong. As a result, there are two thresholds to be designed in such a system. By carefully designing the two thresholds, the feedback is more meaningful and useful for learning.

2. Cost-based Bi-threshold Verification

In the design of a CALL application, one starts with a specification of its *false acceptance rate* (FA) and *false rejection rate* (FA). For instance, a more forgiving CALL system will have a relatively lower FR at the expense of a higher FA. In our bi-threshold CALL system, we have 3 kinds of response: good, fair, and bad pronunciations. We will call the two involving thresholds "good-fair" (GF) threshold and "bad-fair" (BF) threshold. The verification performance on correct and incorrect inputs is then described by 6 probabilities as shown in Fig 1: P(bad|correct), P(fair|correct), P(good|correct), P(bad|incorrect), P(fair|incorrect), and P(good|incorrect). In fact, P(bad|correct) and P(good|incorrect) are supposed to be FR and FA respectively.

Traditionally, one would determine the two required thresholds independently according to the specified FA and FR. However, since the distributions of confidence scores for correct and incorrect inputs are not uniform, if one is allowed to deviate from the specified FA and FR slightly, better performance can actually be achieved by a joint estimation of the two thresholds.

2.1. Cost Function

Let us denote the FA and FR specification of the system as $\widehat{FA}$ and $\widehat{FR}$ respectively. The two thresholds — GF and BF thresholds — are jointly estimated by considering the following three factors:

(I) P(good|correct) and P(bad|incorrect) should be as large as possible.

(II) P(bad|correct) and P(good|incorrect) are basically the FR and FA; they should be as small as possible.

(III) P(bad|correct) and P(good|incorrect) are allowed to deviate from the specified $\widehat{FR}$ and $\widehat{FA}$ but only if they are smaller then the latter; otherwise, they will be penalized.

Intuitively, good thresholds will try to maximize (I) but to minimize (II) and (III). Formally, the estimation of the two thresholds is to maximize the following cost function C:

$$\begin{aligned}&\{k_1 \cdot P(good|correct) + k_2 \cdot P(bad|incorrect)\}\\ &-\{k_3 \cdot P(bad|correct) + k_4 \cdot P(good|incorrect)\}\\ &-\{k_5 \cdot u(x) \cdot x + k_6 \cdot u(y) \cdot y\} \qquad (1)\end{aligned}$$

where k_1 — k_6 are positive constants whose values depend on their importance in the system design; and

$$x = P(good|incorrect) - \widehat{FA} \qquad (2)$$
$$y = P(bad|correct) - \widehat{FR} \qquad (3)$$

and u is the unit step function defined as

$$u(z) = \begin{cases} 1 & \text{if } z > 0.0 \\ 0 & \text{if } z \leq 0.0 \end{cases} .$$

Table 2: Illustration of the cost function. (Inc = Incorrect; Cor = correct)

Case	Class	P(bad)	P(fair)	P(good)	(I) - (II)	(III)	C
1	Inc	0.4	0.4	0.2	0.6	0	0.6
	Cor	0.1	0.4	0.5			
2	Inc	0.65	0.15	0.2	0.65	0.2	0.45
	Cor	0.3	0.2	0.5			
3	Inc	0.48	0.32	0.2	0.65	0.03	0.62
	Cor	0.13	0.37	0.5			

2.2. An Example

Three scenarios are constructed in Table 2 to illustrate the effects of the three factors. Here we assumed that the required $\widehat{FA}$ and $\widehat{FR}$ are 0.2 and 0.1 respectively, and all the constants k_1 — k_6 are 1.0. In case 1, the BF and GF thresholds are designed according to the exact $\widehat{FA}$ and $\widehat{FR}$ values. In case 2, the BF threshold is shifted to the "fair" class so that more instances are now classified as bad. As a result, FR = P(bad|correct) also increases from 0.1 to 0.3, which is too large to be acceptable. However, if one just looks at the score of (I) - (II) then case 2 is actually preferred. The example explains the importance of the factor (III) which acts as a penalty for deviation of the FA and FR from the specified values. The cost function constructed gives a lower score for case 2 than case 1. On the other hand, in case 3, the BF threshold is also shifted to the "fair" class, but the deviation is so small that the increase in P(bad|incorrect) more than compensates the increase in FR.

3. Experimental Evaluation

The cost-based bi-threshold verification algorithm is evaluated in a CALL system we help develop with the Hong Kong Applied Science and Technology Research Institute Company Limited (ASTRI). A speech corpus is collected by ASTRI from local Hong Kong Chinese and English native-speaking kids of Grade 3 to 6. All collected data are phonemically transcribed. The corpus is divided into 2 disjoint sets: training set and evaluation set. The training set contains 13,049 utterances from 153 kids (89 boys, 64 girls). All of them are sentences. The evaluation set consists of 7,381 utterances representing 37,026 phoneme segments from 52 kids (26 boys and 26 girls). Among the 37,026 phoneme segments, 29,257 of them are pronounced correctly while 7,769 of them are pronounced incorrectly.

3.1. Setup

3.1.1. Acoustic Models

There are 40 phonemes in our system. Position-dependent hidden Markov models (HMM) are used to represent the phonemes depending on their position in the beginning, middle, or at the end of a word. Thus, there are totally 120 HMMs. Each HMM has 3 states with 10 Gaussian components per state. They are trained using the HTK Toolkit [6]. The standard 39-dimension acoustic vector is used, consisting of 12 MFCCs, normalized frame energy and their first- and second-order derivatives.

3.1.2. Confidence Measure

Given a phoneme segment X of duration T frames, the confidence score, $CS(C_j|X)$, of X being the j-th phoneme C_j among N phoneme candidates is defined as the time-normalized log-likelihood of its posterior probability $P(C_j|X)$. That is,

$$CS(C_j|X) = \frac{1}{T}\log P(C_j|X) \tag{4}$$

where the posterior probability is defined as

$$P(C_j|X) = \frac{P(X|C_j)P(C_j)}{\sum_{k=1}^{N} P(X|C_k)P(C_k)} . \tag{5}$$

If the prior probabilities of all the phonemes may be assumed equal, then Eqn(5) may be simplified as

$$P(C_j|X) \approx \frac{P(X|C_j)}{\sum_{k=1}^{N} P(X|C_k)} . \tag{6}$$

We follow [5] and compute the denominator from its Viterbi path. That is,

$$P(C_j|X) \approx \frac{P(X|C_j)}{max_{1\le k\le N} P(X|C_k)} . \tag{7}$$

However, since Eqn(7) involves all possible phonemes, it requires substantial computational cost. To speed up, we further approximate the denominator using a Gaussian mixture model (GMM) trained on all phoneme data. The GMM has 32 mixture components.

To summarize, the verification procedure runs as follows: A user is asked to utter a prompted text. The user's utterance is then forced-aligned with the known phonemic transcription of the text. A Viterbi search is also performed with the GMM trained from all phonemes. For each aligned phoneme segment, the confidence score of Eqn(4) is computed, and according the the two thresholds — GF and BF thresholds — it is classified as good, fair, or bad.

3.1.3. Use of Margin

Using the cost function defined in Eqn(1), the GF and BF thresholds are found by exhaustive search. In the determination of the thresholds, we always find the ones with the largest margin. For example, if there is a range of thresholds that will gives 10% FA, we always take the midpoint of that range in order to achieve the minimum risk (or better generalization) on testing sets.

3.1.4. Evaluation Metrics

Two metrics are used for evaluation:

- the verification error defined as the sum of P(bad|correct) and P(good|incorrect).
- the cost defined by Eqn(1).

Ten-fold cross-validation is performed. That is, the evaluation data set is divided into 10 subsets. The two cost-based thresholds are estimated from 9 of the 10 subsets and then tested on the hold-out subset; the experiment is repeated 10 times by cycling the hold-out subset through the 10 available subsets. Mean and standard deviation of results from the 10 cross-validation are reported.

3.1.5. Miscellaneous

In this evaluation, all the 6 positive constants k_1—k_6 in Eqn(1) are set to 1.0. For phoneme-class thresholds, the 40 phonemes are categorized into 9 phoneme classes: affricates, diphthongs, fricatives, nasals, semi-vowels, stops, back vowels, middle vowels, and front vowels.

Table 3: Results from thresholds determined from exact $\widehat{FA}$ and $\widehat{FR}$.

Threshold Type	Statistic	Error	Cost
per-phoneme	mean	0.3342	0.4698
	std dev	0.0372	0.0355
phoneme-class	mean	0.3058	0.5579
	std dev	0.0375	0.0327

3.2. Experiment I: Exact FA and FR

Table 3 shows the results when the two thresholds are determined independently according to the exact $\widehat{FA}$ and $\widehat{FR}$ requirements. The result using phoneme-class thresholds has lower verification error (by about 3% absolute) and higher cost value (by about 9%) than if per-phoneme thresholds are used. This is mainly due to the problem of insufficient data to estimate each phoneme threshold robustly. The phoneme-class thresholds also give a more stable cost.

Table 4: Results from thresholds determined by maximization of the cost function in Eqn(1).

Threshold From	Statistic	Error	Score
Exact $\widehat{FA}/\widehat{FR}$	mean	0.3058	0.5579
	std dev	0.0375	0.0327
Cost-based	mean	0.2985	0.6084
	std dev	0.0252	0.0314

3.3. Experiment II: Cost-based Thresholds

Table 4 compares the performance using thresholds determined from exact $\widehat{FA}$ and $\widehat{FR}$ requirements and thresholds estimated using the cost-based algorithm described in Section 2. Both verification performance and the defined cost are improved. The cost-based thresholds give more stable performance (with smaller variances) too.

Table 5: Effect of bagging on cost-based bi-threshold verification

Bagging?	Statistic	Error	Cost
No	mean	0.2985	0.6084
	std dev	0.0252	0.0314
Yes	mean	0.2835	0.6402
	std dev	0.0309	0.0314

3.4. Experiment III: Effect of Bagging

We also investigate the use of *bagging* (bootstrap aggregation) to improve the robustness of the linear thresholds. Bagging is commonly used, especially in the area of *ensemble of classifiers* [3], to enhance performance stability. The results in Table 5 show that the bagging technique can further improve the mean performance, both in terms of verification error (up by about 1%) and the cost (up by about 4%). However, we do not find additional stability in the performance.

4. Conclusions

In this paper, we present a novel approach to enhance the usability of a verification-based application when the verification reliability is not high. We propose to re-cast the problem into a 3-class verification problem with the creation of an additional "in-between" class. One such application is in the area of building a CALL system. Furthermore, if one may relax the false acceptance rate and false rejection rate requirements slightly, the two thresholds in a CALL system may be jointly estimated by maximization of a cost function. Compared with a system using per-phone thresholds determined from exact FA and FR, the use of phoneme-class thresholds estimated from a cost function that relaxes the FA and FR slightly together with the bagging technique reduces verification error from 33.4% to 28.4%. The performance stability also improves from a standard deviation of 0.0375 to 0.0309.

5. Acknowledgements

This work is supported by the Hong Kong Applied Science and Technology Research Institute Company Limited.

6. References

[1] Tasuo Ariki and Jun Ogata, "English CALL System with Functions of Speech Segmentation and Pronunciation Evaluation Using Speech Recognition Technology", ICSLP 02, 2002.

[2] C. Cucchiarini, H. Strik, and L. Boves, "Automatic Evaluation of Dutch Pronunciation by Using Speech Recognition Technology", Proc. of IEEE ASRU, Santa Barbara, Dec. 1997.

[3] T.G. Dietterich, "Ensemble Methods in Machine Learning", Proc. of Multiple Classifier Systems, June 2000.

[4] K. Y. Leung and M. H. Siu, "Phone Level Confidence Measure Using Articulatory Features", ICASSP 03, 2003.

[5] S. Witt and S. Young, "Language Learning Based on Non-native Speech Recognition", Eurospeech 97, 1997.

[6] S. Young, D. Kershaw, J. Odell, D. Ollason, V.Valtchev and P.Woodland, "The HTK Book for HTK 3.0", Microsoft Corporation, July 2000.