AN ALTERNATIVE APPROACH OF FINDING COMPETING HYPOTHESES FOR BETTER MINIMUM CLASSIFICATION ERROR TRAINING

Yik-Cheung Tam and Brian Mak

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

ABSTRACT

During minimum-classification-error (MCE) training, competing hypotheses against the correct one are commonly derived by the N-best algorithm. One problem with the N-best algorithm is that, in practice, some misclassified data can have very large misclassification distances from the N-best competitors and fall out of the steep/trainable region of the sigmoid function, and thus cannot be utilized effectively. Although one may alleviate the problem by adjusting the shape of the sigmoid and then using an appropriate learning rate, it requires careful tuning of these training parameters. In this paper, we propose using the nearest competing hypothesis instead of the traditional N-best hypotheses for MCE training. The aim is to keep the training data as close to the trainable region as possible. Consequently, the amount of "effective" training data is increased. Furthermore, by progressively beating the nearest competitors, the training seems to be more stable. We also design an approximation algorithm based on beam search to locate the nearest competing hypothesis efficiently. We compare the performance of MCE training using 1-nearest or 1-best competing hypotheses on the Aurora database and find that the new approach (using 1-nearest hypotheses) reduces the word error rates by 5.1% and 17.8% over the latter (of using the 1-best competing hypotheses) and the official Aurora baseline respectively.

1. INTRODUCTION

Minimum classification error (MCE) training is a powerful discriminative technique to optimize any system parameters so that the ultimate classification errors are minimized. It has been successfully applied to various problems in speech recognition; for instance, optimizing the hidden Markov model (HMM) parameters [1, 2], discriminative feature extraction [3], and finding optimal subband weightings [4].

MCE training involves two steps to establish the optimization criterion. Firstly, a misclassification distance measure $d(X)$ is defined. It represents the "average" distance of competing hypotheses from the correct hypothesis for an utterance X as follows:

$$d(X) = \log \left\{ \frac{1}{N} \sum_{n=1}^{N} exp(\eta \cdot g_n) \right\}^{\frac{1}{\eta}} - g_0 \qquad (1)$$

where $g_j, 0 < j \leq N$ denotes a set of N discriminant functions for each of the N competing hypotheses; and, g_0 denotes the one for the correct hypothesis. g_j is commonly computed as the Viterbi log-likelihood of the j-th competing hypothesis. Competing hypotheses can be derived using the N-best algorithm [5, 6]. Secondly, a misclassification distance is turned into a soft error count. The 0-1 sigmoid function is usually employed:

$$l(d(X)) = \frac{1}{1 + \exp(-\gamma d(X) + \alpha)} \qquad (2)$$

where γ and α control its shape and offset respectively.

Given the training set $\{X_i, 1 \leq i \leq U\}$, the empirical recognition error for minimization is given by

$$R_{mce} = \frac{1}{U} \sum_{i}^{U} l(d(X_i)) \, . \qquad (3)$$

Using the iterative generalized probabilistic descent (GPD) algorithm [1], a better parameter estimate θ may be obtained by the following update rule:

$$\theta(t+1) = \theta(t) - \epsilon(t) \left. \frac{\partial R_{mce}}{\partial \theta} \right|_{\theta=\theta(t)} \qquad (4)$$

where $\epsilon(t)$ is the learning rate at the t-th iteration.

We observe that an utterance X is hardly trainable if its misclassification distance is so large that it falls outside the steep region of the sigmoid function, and its $l(d(X))$ approaches one. In light of this, instead of using the N-best competitors to compute the misclassification distance, we propose using the *nearest* competitor. By definition, under the same setup of corrective MCE training, the 1-nearest competitor always has a misclassification distance smaller than those of the N-best counterparts (except when only one competitor is better than the correct hypothesis; in that case, $N = 1$ and they are the same). Thus, the 1-nearest competitors are always closer to the trainable region of the sigmoid function; or in other words, one will have more "effective" training data using 1-nearest competitors than using N-best competitors. This can be important when the baseline recognition performance is already quite good and there will be very few training data for MCE training. Another viewpoint is that, intuitively, if the best competitor is so strong that it cannot be beaten directly, it may be better to beat the *weaker* nearest competitors progressively and hopefully it can find its way up to the strongest competitor. Figure 1 shows the distributions of the soft error counts of the 1-best and the 1-nearest competitors during the 1st iteration of MCE training of the Aurora corpus (see Section 3.3.1). From the figure, it can be seen that most 1-best competing hypotheses concentrate in the

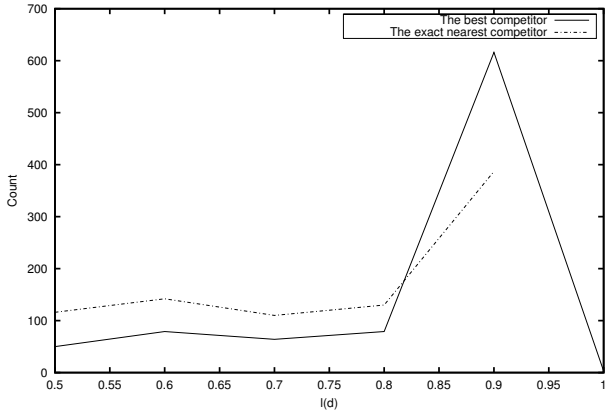

Fig. 1. Distributions of the soft error counts of the 1-best and the 1-nearest competitors on the Aurora corpus during the first iteration of MCE training. The sigmoid slope is 0.1.

region of large $l(d)$ values: $0.8 \leq l(d) \leq 1.0$, while there are many more 1-nearest than 1-best competitors for $l(d) < 0.8$.

In this paper, we will show, through a series of experiments, that it is preferable to use 1-nearest competing hypotheses than 1-best competitors for MCE training. We also have designed an approximation algorithm to compute the 1-nearest hypothesis that is more efficient in time and space.

2. ALGORITHMS TO LOCATE THE NEAREST COMPETITOR

In this section, we will introduce two algorithms to locate the nearest competitor. The first algorithm will find the exact 1-nearest competitor based on the N-best algorithm. The second one is an approximation algorithm based on beam search.

2.1. The Exact Algorithm

The algorithm is outlined in pseudo-codes as follows:

INITIALIZE:
i = 1 and S_0 = null hypothesis
LOOP
Retrieve the next competing hypothesis S_i using the N-best algorithm.
IF S_i matches the correct hypothesis **THEN**
RETURN S_{i-1}
END LOOP

The advantage of this approach is that it can accurately locate the 1-nearest competitor by scanning through the rank-ordered N-best list in a top-down fashion. However, in case of an extremely low-ranking 1-nearest competitor, this approach will take a long time and may run into a memory problem. In practice, one has to limit the search depth whose value is application dependent. In this paper, we set the maximum search depth to 1000.

2.2. The Approximation Algorithm

The algorithm is a slight modification of the Viterbi algorithm. The forced-alignment path of the correct hypothesis is taken as a reference and Viterbi decoding is performed with the constraint that

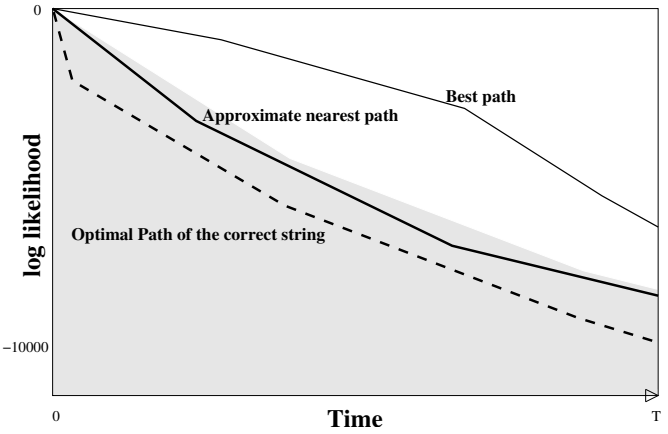

Fig. 2. The search space of an approximation algorithm that locates the 1-nearest competitor

any partial paths with an accumulated likelihood *greater* than the accumulated likelihood of the reference at time t, $V_{ref}(t)$, by a beam-width of $beam(t)$ are pruned. The idea is illustrated in Figure 2 in which the shaded region represents the valid search space of this approximation algorithm. Let $V(t,j)$ and $b_t(j)$ denote the accumulated likelihood and the state observation likelihood of the HMM state j at time t respectively. Then our Viterbi updating formula at time $(t+1)$ is modified as follows:

$$V(t+1,j) \quad = \max_i \{V(t,i) \cdot a_{ij} \cdot b_{t+1}(j)\} \tag{5}$$

where $V(t,i) \cdot a_{ij} \cdot b_{t+1}(j) \leq V_{ref}(t+1) \cdot beam(t+1)$.

Since the reference path of the correct hypothesis is *optimal*, and the output path of the algorithm must be better than the reference path given a suitable beam-width, a competing hypothesis will be returned. The beam-width controls how close the output competing hypothesis is to the correct hypothesis. If the beam-width is too small, the output degenerates to the reference path. On the other hand, if the beam-width tends to infinity, the best hypothesis will be returned. In our experiments, we started with an initial beam-width of 10, and if the output fell back to the reference hypothesis, we incremented the beam-width by 10 progressively until we found an approximate 1-nearest hypothesis. In most cases, an approximate 1-nearest competitor could be found within the preset beam-width of 10. The approximation algorithm solves both the memory and computation problem of the exact approach.

3. EXPERIMENTS AND DISCUSSION

The performance of MCE training using

- the best competitor;
- the nearest competitor; or,
- the approximate nearest competitor

was investigated using the Aurora corpus [7]. Resulting models were evaluated on the test set A which matches the noise types and channel characteristics of the training set.

3.1. The Aurora Corpus

The Aurora corpus was created to support distributed speech recognition research under noisy environments. Continuous TIDIG-

Table 1. The best word error rates on the test set A among the first 3 MCE/GPD iterations with different γ and $\epsilon(0) = 422$.

System	γ=0.5	γ=0.1	γ=0.02	γ=0.004
baseline(ML)	12.71%	12.71%	12.71%	12.71%
1-best	11.55%	**11.01%**	11.08%	12.07%
approx-1-nearest	10.85%	**10.71%**	11.27%	12.27%
exact-1-nearest	10.46%	**10.45%**	10.92%	12.16%

Table 2. String error rates on the training set during the first 3 MCE/GPD iterations. $\gamma = 0.1$ and $\epsilon(0) = 422$.

System	Before MCE	1 Iter	2 Iter	3 Iter
1-best	10.68%	9.00%	8.31%	7.46%
approx-1-nearest	10.68%	8.92%	8.19%	7.32%
exact-1-nearest	10.68%	8.68%	7.90%	7.29%

ITS [8] were first pre-filtered according to the frequency characteristics of common telecommunication channels (G.712 or MIRS). Then, realistic noises were artificially added at 6 different signal-to-noise (SNR) ratios ranging from 20dB to -5dB at 5dB steps for the test sets and from 20dB to 5dB at 5dB steps for the multi-condition training set. Two training modes: clean training and multi-condition training, and three test sets are defined to evaluate recognition performance subject to matched/unmatched noises, and matched/unmatched channel characteristics.

3.2. Experimental Setup

We followed the exact procedure in the official Aurora paper [7] to create our maximum-likelihood (ML) baseline using HTK. The feature vector consisted of 12 MFCCs and the logarithmic frame energy + Δ + $\Delta\Delta$. Each digit was represented by a whole-word HMM, which was strictly left-to-right, with 16 states and 3 Gaussian mixtures per state. The silence model was a 3-state HMM with 6 Gaussian mixtures per state. The short pause with skip is a 1-state HMM that was tied with the 2nd state of the silence model. All models were trained by the Baum-Welch algorithm to obtain the initial ML models. Corrective string-based MCE training was then performed, and the learning rate was decreased along training iterations t by the following formula: $\epsilon(t) = (1 - t/50) \cdot \epsilon(0)$ with t $\geq$ 0 and we limited the maximum number of iterations to 50. Furthermore, the sigmoid offset was set to zero in all our experiments.

3.3. Experiment I: The Effect of the Sigmoid Slope

The aim of this experiment is to investigate the impact of different steepness of the sigmoid function to the training performance by using different γ: $\{0.5, 0.1, 0.02, 0.004\}$. MCE training was performed for three iterations and the initial learning rate $\epsilon(0)$ was set to 422 which was empirically found to give good results.

3.3.1. Case I: With $\gamma = 0.1$

This gamma value yields the best performance among all the four gamma values tested as shown in Table 1. MCE training with the exact nearest competitor reduces the word error rate by 5.1% and 17.8% compared with the one using the best competitor and the ML baseline respectively. As expected, the performance of MCE training using the approximate nearest competitor is better than that using the best competitor but worse than that using the exact nearest competitor. The new approach using the exact nearest competitor also demonstrates a faster convergence than the one using the 1-best competitor as shown in Table 2.

Table 3. Percentage of effective training data. Data with $l(d(X)) < 0.95$ are defined to be effective.

System	γ=0.5	γ=0.1	γ=0.02	γ=0.004
1-best	9.43%	39.73%	94.39%	100%
approx-1-nearest	12.53%	51.03%	97.61%	100%
exact-1-nearest	20.23%	67.01%	99.44%	100%

Table 4. The best word error rates on the test set A within the first 3 MCE/GPD iterations. $\gamma = 0.004$ and $\epsilon(0) = 10550$.

System	Overall word error rates
baseline(ML)	12.71%
1-best	11.55%
approx-1-nearest	10.70%
exact-1-nearest	10.79%

3.3.2. Case II: With larger slope, $\gamma = 0.5$

We investigated the effect of using a sigmoid function with a slope steeper than the (empirically) optimal value of 0.1. The results are tabulated in Table 1. In terms of word error rate, the performance with the use of the best competitors deteriorates by 4.9% from its corresponding result that uses $\gamma = 0.1$. On the other hand, the performance degradation of MCE training using the approximate nearest competitors is 1.3% and MCE training using exact 1-nearest competitors does not degrade at all. It seems that MCE training using the nearest competitors is less sensitive to an increase in γ when compared with that using 1-best competitors. This may be explained by the fact that more effective amount of training data is available for the new approach according to Table 3.

3.3.3. Case III: With smaller slopes, $\gamma = \{0.02, 0.004\}$

From Table 1, when the slope of the sigmoid function becomes flatter, the advantage of using the nearest competitor disappears. To illustrate this, we compute the effective amount of training data in all three approaches during the first MCE/GPD iteration in Table 3, which is defined as the amount of data whose soft errors, $l(d(X))$, are less than 0.95. Notice that with a slope of 0.004, the sigmoid is so flat that all training data fall within the trainable region in all three methods. Thus, it is better to use the best competitors *directly* rather than the nearest competitors. Nevertheless, their performance difference is small. However, because of the small slope, convergence is slow and the MCE training had not converged in 3 iterations.

3.4. Experiment II: Compensation on Using a Flat Sigmoid

In Experiment I, we find that the sigmoid slope determines the amount of "effective" training data which, in turn, affects the relative performance among the three methods. It is interesting to

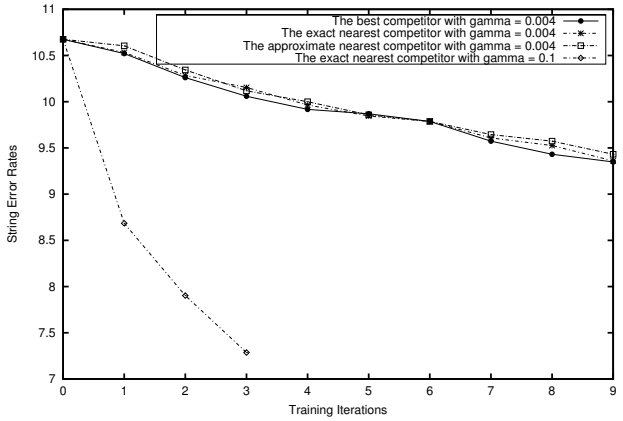

Fig. 3. String Error Rates on the training set. $\gamma = 0.004$, $\epsilon(0) = 422$.

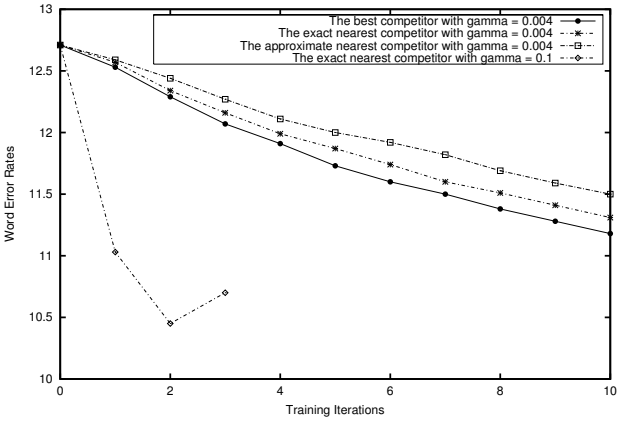

Fig. 4. Word error rates on the test set A. $\gamma = 0.004$, $\epsilon(0) = 422$.

compare the three methods under the condition that they have the same amount of effective training data. However, a small slope results in slow convergence of MCE training. In this experiment, we choose a flat sigmoid with $\gamma = 0.004$, and investigate the performance of MCE using the three kinds of competitors when training converges.

3.4.1. Case I: Using a larger learning rate

We followed the same procedure as in Experiment I except that we increased the initial learning rate $\epsilon(0)$ to 10550 so that the product, $\gamma \times \epsilon(0)$, is the same as in the best case with $\gamma = 0.1$ and $\epsilon(0) = 422$. The motivation is to compensate for the slow learning due to the flat sigmoid. Results are tabulated in Table 4. After the compensation, the results are comparable to the best results in Table 1 of Experiment I. However, the approach using the nearest competitors is still better than the one using the best competitor. One possible reason is that overshoot occurs due to the large learning rate.

3.4.2. Case II: Applying more training iterations

We also allowed more training iterations with the same learning rate of 422 as in Experiment I and checked its convergence while avoiding training overshoot. When compared with MCE training using a steep sigmoid slope of $\gamma = 0.1$, the convergence is very slow as indicated by the the slow decrease in string error rates on the training set in Figure 3. Despite more training iterations, recognition performance is not as good as the best case in Experiment I according to the results in Figure 4. Therefore, such setting is not practical since many more training iterations are required.

4. CONCLUSION

In this paper, we propose using the nearest competing hypotheses for MCE training. We also present an approximation algorithm which, although, is not guaranteed to find the exact 1-nearest hypothesis, is much faster and requires much less memory. On the Aurora task, in the best case, MCE training using exact 1-nearest competitors achieves a reduction in word error rate of 5.1% and 17.8% compared with that using 1-best competitors and the official Aurora baseline respectively. We investigated the effect of the amount of "effective" training data by varying the slope of the sigmoid function to show that the 1-nearest method always results in the greatest amount of effective training data as expected and gives the best performance in a few MCE training iterations except when the amount of effective data is the same for all three methods. In the latter case, we show that even with many more training iterations, training with 1-best competitors still converges too slowly to compare with the best result using 1-nearest competitor. On the other hand, if the convergence is sped up by a larger learning rate, training overshoot may occur. In summary, MCE training using 1-nearest competing hypotheses seems to outperform that using 1-best competing hypotheses in various settings of the training conditions with a faster convergence.

5. REFERENCES

[1] B.H. Juang, W. Chou, and C.H. Lee, "Minimum Classification Error Rate Methods for Speech Recognition," *IEEE Trans. on SAP*, vol. 5, no. 3, pp. 257–265, May 1997.

[2] W. Chou, C.H. Lee, and B.H. Juang, "Minimum Error Rate Training Based on N-best String Models," in *Proceedings of ICASSP*, 1993, vol. 2, pp. 652–655.

[3] A. Biem, S. Katagiri, E. McDermott, and B.H. Juang, "An Application of Discriminative Feature Extraction to Filter-Bank-Based Speech Recognition," *IEEE Trans. on SAP*, vol. 9, no. 2, pp. 96–110, Feb 2001.

[4] Y. C. Tam and B. Mak, "Development of an Asynchronous Multiband System for Continuous Speech Recognition," in *Proceedings of Eurospeech*, 2001, vol. 1, pp. 575–578.

[5] F. K. Soong and E. F. Hwang, " Tree-Trellis Based Fast Search for the Finding of the N-best Sentence Hypotheses in Continuous Speech Recognition," in *Proceedings of ICASSP*, 1991, pp. 705–708.

[6] R. Schwartz and Y. L. Chow, "he N-best Algorithm: An Efficient and Exact Procedure for Finding the N Most Likely Sentence Hypotheses," in *Proceedings of ICASSP*, 1990, pp. 81–84.

[7] H. G. Hirsch and D. Pearce, "The AURORA Experimental Framework for the Performance Evaluations of Speech Recognition Systems under Noisy Conditions," in *ISCA ITRW ASR2000 "Automatic Speech Recognition: Challenges for the Next Millennium"*, September 2000.

[8] R.G. Leonard, "A Database for Speaker-Independent Digit Recognition," in *Proceedings of ICASSP*, 1984.