STRING-LEVEL MCE FOR CONTINUOUS PHONEME RECOGNITION

Erik McDermott Shigeru Katagiri
ATR Human Information Processing Res Labs
2-2 Hikari-dai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan

ABSTRACT

In this paper, we present results for the Minimum Classification Error (MCE) [1] framework for **discriminative training** applied to tasks in **continuous phoneme recognition**. The results obtained using MCE are compared with results for Maximum Likelihood Estimation (MLE). We examine the ability of MCE to attain high recognition performance with a small number of parameters. Phoneme-level and string-level MCE loss functions were used as the optimization criteria for a Prototype-Based Minimum Error Classifier (PBMEC) [2] and an HMM [3]. The former was optimized using Generalized Probabilistic Descent, the latter was optimized using an approximated second order method, the Quickprop algorithm. Two databases were used in this evaluation: 1) the ATR 5240 isolated word datasets for 6 speakers, in both speaker-dependent and multi-speaker mode; 2) the TIMIT database. For both databases, MCE training yielded striking gains in performance and classifier compactness compared to MLE baselines. For instance, through MCE training, performance similar to that of the Maximum Likelihood Successive State Splitting algorithm (ML-SSS) [4] could be obtained with 20 times fewer parameters.

1. INTRODUCTION

The MCE framework was first proposed in [1]. The essence of MCE optimization is that the loss function is a smooth approximation of the actual classification error. Through minimization of this loss function, MCE focuses directly on minimizing classification error rather than on learning the true data probability distributions (the target of MLE). This can allow a classifier to achieve high performance, even with a small number of parameters.

Practical application of this framework to specific classifiers has been described for hidden Markov models [3] and prototype-based classifiers [2] [5]. These studies report clear improvements in isolated word recognition for MCE compared to MLE based classifier design. Additional studies showed that minimizing a string-level MCE loss is a practical approach to optimizing continuous word accuracy [6] [7]. The string-level loss incorporates the connected word Dynamic Programming procedure in a readily optimizable functional form.

Using discriminative training to achieve high performance continuous phoneme recognition *without* the use of lexical constraints is a practical way of generating accurate phoneme models that are suited to general tasks in speech recognition. In this report we evaluate phoneme-level and string-level MCE training for tasks in continuous phoneme recognition, on the ATR isolated word database and on the TIMIT database. Continuous phoneme recognition on the ATR isolated word database was used to evaluate ML-SSS [4], and continuous phoneme recognition on the TIMIT database has been used to evaluate a number of recognizer structures, including recurrent neural networks, context-dependent HMMs, and HMMs optimized using Maximum Mutual Information (MMI) [8, 9].

2. RECOGNIZER DESIGN

We made use of two recognizer structures: the Prototype Based Minimum Error Classifier (PBMEC) [2] and an HMM, both trained using MCE. PBMEC closely resembles an HMM in structure, but is not explicitly probabilistic.

2.1. Discriminant functions for phoneme/string categories

Discriminant functions are defined both for 1) phoneme categories given a labeled speech segment and 2) string categories given an entire utterance. String categories are defined by a connected phoneme grammar used to constrain the matching of the utterance to the acoustic models of the recognizer. String categories consist of sequences of phonemes allowed by the grammar. Phoneme categories in the context of labeled speech segments can be viewed as string categories in a restricted grammar, defined over a mini-utterance (i.e., the labeled segment). Hence, in this and the following section, the MCE formalism is described for general string categories, with the understanding that phoneme-level MCE is a special case of string-level MCE.

Given an utterance represented as a sequence of feature vectors, $\mathbf{x}_1^T = (\mathbf{x}_1, ..., \mathbf{x}_t, ..., \mathbf{x}_T)$, the best DP sequence of sub-phonemic states $\Theta = (\theta_1, ..., \theta_t, ..., \theta_T)$ for string category C_j is used to define the PBMEC discriminant function for C_j:

$$g_j(\mathbf{x}_1^T, \Lambda) = \sum_{t=1}^{T} e_{\theta_t}(\mathbf{x}_t), \qquad (1)$$

where $e_i(\mathbf{x}_t)$ is the local PBMEC state distance between a feature vector and the reference vectors (with associated covariance matrix) of PBMEC state i. Similarly, the best Viterbi state sequence for string category C_j is used to define the HMM discriminant function for C_j:

$$g_j(\mathbf{x}_1^T, \Lambda) = \sum_{t=1}^{T} \log a_{\theta_{t-1}\theta_t} + \sum_{t=1}^{T} \log b_{\theta_t}(\mathbf{x}_t), \qquad (2)$$

where a_{ij} denotes a transition probability from state i to state j, and $b_i(\mathbf{x}_t)$ denotes the observation probability of a feature vector in state i.

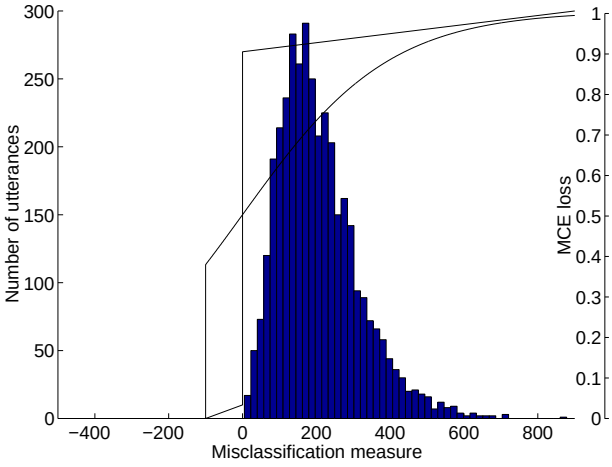

Figure 1. String-level misclassification measure distribution for the TIMIT training set before MCE optimization, for a 3-state, 16-component HMM; two possible MCE loss functions.

2.2. String-level MCE misclassification measure and loss function

A misclassification measure compares the correct string with the incorrect strings. For HMMs, the following measure was used:

$$d_k(\mathbf{x}_1^T, \Lambda) = -g_k(\mathbf{x}_1^T, \Lambda) + \log\left[\frac{1}{M-1}\sum_{j\neq k}^{M} e^{g_j(\mathbf{x}_1^T,\Lambda)\psi}\right]^{\frac{1}{\psi}}. \quad (3)$$

M denotes the number of possible strings. For a large ψ, the bracketed expression is approximately the value of the discriminant function of the best incorrect category.

The usual smoothed MCE loss function is then used to define a **string-level loss** for a given utterance, denoted $\ell(\mathbf{x}_1^T, \Lambda)$. Typically, the MCE loss function is asymmetric: the derivative should be significant for most incorrectly classified training tokens, but not for most correctly classified tokens. The steepness of the loss function must be chosen appropriately given the (task-dependent) range of the misclassification measure; this is illustrated in Figure 1 with two typical loss functions, a piece-wise linear function and a chopped sigmoid.

The total loss $L(X, \Lambda)$ is the local loss summed over the training data $X = X_1, ..., X_m$, where each X_n is a sequence of feature vectors, i.e. $X_n = \mathbf{x}_{n,1}^{T(n)}$. The optimization of this loss, for both PBMEC and HMM, is described below.

2.3. Contrast between string-level MCE and phoneme-level MCE

The phoneme-level loss assumes that label information is available. Clearly, this information is not available for unseen data. In this light, the string-level MCE loss is defined in a manner that is closer to the final test of recognition in which label information is not available and DP has to be used to find the best segmentation and best phoneme sequence.

Though use of a string-level MCE loss is directly geared at minimizing string-level errors, it has the effect of reducing phoneme errors too. String-level MCE focuses on the differences between correct and best incorrect strings in a way that is intuitively related to minimizing substitutions, deletions and insertions. This is illustrated in Figure 2.

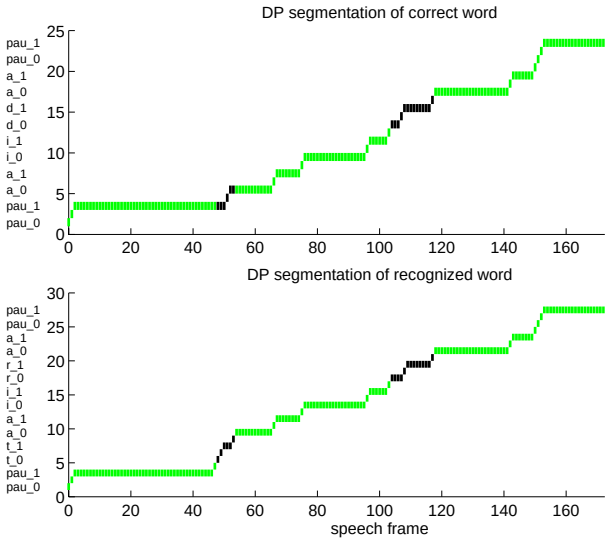

Figure 2. The gradient of the string-level loss function cancels out for the phonemes found in both correct and best incorrect strings. Training focuses on the regions where there is a difference between correct and incorrect DP segmentations.

In this light, phoneme-level MCE seems less appropriate than string-level MCE when the target is continuous phoneme recognition. However, phoneme-level training may be more effective at **separating** correct from incorrect phonemes. In string-level MCE, if only the single best incorrect string category is used in the definition of misclassification measure (Equ. (3)), no learning occurs for most correctly recognized phonemes: the gradient cancels out. In contrast, since phoneme-level MCE compares correct and best incorrect phonemes for every segment, learning occurs for some correctly recognized segments as well as for most incorrectly recognized segments, depending on the form of the loss function. Phoneme-level training will thus attempt to increase the separation between correct and best-incorrect phonemes. This can have a large effect on the robustness of the classifier. A similar "separating" effect can be achieved for string-level MCE only if many top incorrect categories are used in the definition of misclassification measure. For long utterances, the number of these categories must be quite large to generate a competing category for every phoneme segment.

Both string-level MCE and phoneme-level MCE are evaluated in the experiments below.

2.4. Gradient based sequential optimization

For the PBMEC models used below, the stochastic, gradient based Generalized Probabilistic Descent (GPD) [1] approach was used to minimize the string-level MCE loss. In GPD, the system parameters Λ are adapted according to

$$\Lambda_{n+1} = \Lambda_n - \epsilon_n \nabla \ell(X_n, \Lambda_n), \quad (4)$$

where ϵ_n is a time-decreasing step size.

This procedure will minimize the overall loss $L(X, \Lambda)$, which is the local loss $\ell(X_n, \Lambda_n)$ summed over the training data $X = X_1, ..., X_m$, where each X_n is a sequence of feature vectors, i.e. $X_n = \mathbf{x}_{n,1}^{T(n)}$.

The calculation of the gradient $\nabla \ell(X_n, \Lambda_n)$ is detailed in [10].

# mix. components	52	104	208	416	1170
speaker MAU					
PBMEC	95.8	96.8	97.2	97.4	98.1
ML-SSS			93.6	95.2	96.7
speaker MHT					
PBMEC	95.9	97.4	97.7		
ML-SSS			93.9	95.4	96.1
speaker MXM					
PBMEC	94.2	95.8	96.0		
ML-SSS			91.9	93.9	95.3
speaker FTK					
PBMEC	95.0	95.8	96.5	96.7	97.6
ML-SSS			91.5	94.0	95.0
speaker FMS					
PBMEC	95.0	95.5	96.4		
ML-SSS			91.3	93.2	94.6
speaker FYM					
PBMEC	95.5	96.8	96.6		
ML-SSS			92.4	93.6	95.5
avg PBMEC	95.2	96.4	96.7	97.0	97.8
avg ML-SSS			92.4	94.2	95.5
Multi-speaker					
PBMEC			91.9	92.6	
ML-SSS			85.1	86.9	89.8

Table 1. Phoneme recognition accuracies for PBMEC (trained with string-level MCE) and ML-SSS

2.5. Second order batch optimization

When designing a classifier using a large body of training data, it is highly desirable that one be able to parallelize the training over many different processors. One disadvantage of stochastic descent is that it is difficult to parallelize. For the HMMs used below, a second order batch optimization approach, the Quickprop algorithm [11] was used. Quickprop is a heuristic approximation of the classic Newton's method that can be run in batch mode and so is easy to parallelize over many machines.

Assuming a positive-definite Hessian matrix, Newton's method uses a second order Taylor expansion to model the function of interest. It then calculates the step size that moves to the minimum of the model. As the model is not the true function, the process is then iterated. In *quasi-Newton* methods, to ensure that the Hessian used in the model is positive-definite, a positive term μI can be added to it [12]. Applied to the overall string-level MCE loss function $L(X, \Lambda_n)$, the Newton step is then:

$$\delta\Lambda_{n+1} = -[\nabla^2 L(X, \Lambda_n) + \mu I]^{-1} \nabla L(X, \Lambda_n). \quad (5)$$

One can view the Quickprop algorithm as a heuristic way of performing this modification to the Hessian, assuming a diagonal model of the Hessian, and placing limits on the step size allowed.

In preliminary experiments on TIMIT, we found that Quickprop was slightly more effective in minimizing the MCE loss than the GPD training method. We do not present a detailed comparison here.

3. EXPERIMENTS

3.1. Continuous phoneme recognition for speaker-dependent / multi-speaker word data using PBMEC

3.1.1. Experimental conditions

We examined six speakers (MHT, MAU, MXM, FTK, FYM, FMS) from the ATR 5240 isolated word database, in both speaker-dependent and multi-speaker mode. Half of the database was used for training, the other half for testing. In multi-speaker mode the training data from the six speakers were grouped into a single body of training data, and likewise for the testing data. These tasks were investigated for the Maximum Likelihood Successive State Splitting (ML-SSS) algorithm [4]. ML-SSS uses a Maximum Likelihood criterion to guide the design of sophisticated hidden Markov model topologies for context-dependent phoneme models.

The utterances were parameterized using the same parameterization used for ML-SSS: LPC cepstrum, power, delta-cepstrum and delta-power (34 coefficients) were computed from a 20 ms Hamming window with a 5 ms shift.

A connected phoneme grammar, with rudimentary phonotactic constraints, was used constrain the DP search during both training and testing. This grammar is slightly less constrained than the one used in [4].

A PBMEC pair of reference vector and covariance matrix is here referred to as a "mixture component". Twenty-six phonemes were modeled using a uniform context-independent topology of 2 states with either 1, 2, 4 or 8 components, or 3 states with 15 components (52, 104, 208, 416 or 1170 components in all, respectively.) Segmental K-means on labeled phoneme data was used to initialize the PBMEC models prior to MCE training. The ML-SSS results are for totals of 200, 400 and 1200 components.

3.1.2. String-level MCE training of PBMEC

The GPD procedure was used to minimize the string-level MCE loss. After 20 iterations, the PBMEC models were then tested on unseen data. Deletions, substitutions and insertions were given the same weight in calculating continuous phoneme recognition accuracy. Phoneme accuracies for PBMEC and the best result for either the original SSS or the newer ML-SSS, for different total numbers of mixture components, are shown in Table 1.

3.2. Continuous phoneme recognition for speaker-independent sentence data using HMMs

3.2.1. Experimental conditions

The experimental conditions for the evaluation of MCE-based HMM design on TIMIT followed those described in [13] for the evaluation of Maximum Mutual Information (MMI) based HMM design. TIMIT speech was parameterized using HCode: 12 MFCC coefficients, log-energy, and the corresponding delta and delta-delta coefficients (39 coefficients in all) were computed at a 10 ms frame rate, using a 25 ms Hamming window. The 48 phoneme set was used for training and the 39 phoneme set for testing. The grammar for the task consists of an unconstrained loop over the 48 phonemes. In contrast with [13], bigram probabilities of phoneme transitions were incorporated into the DP search during testing only. A bigram scaling factor was chosen appropriately for each configuration.

3.2.2. String-level MCE training of HMMs

String-level MCE was used with 15-30 iterations of Quickprop to train 3-state context-independent HMMs, initialized using MLE based Viterbi training, on the 3696 TIMIT training sentences, for various numbers of mixture components per state. Two settings, 4 and 20, were investigated for the number of incorrect strings in the string-level MCE misclassification measure (Equ. (3)). Both choices yielded very similar results. Continuous phoneme recognition accuracy (with substitutions, insertions and deletions given the same weight) on the 192 core

components/state	MLE	MCE
1	57.5	61.0
4	64.8	66.2
8	66.9	67.4
16	68.0	68.8

Table 2. Continuous phoneme recognition accuracies for MLE and string-level MCE.

components/state	classification	recognition
1	71.4	62.6
4	75.7	67.5
8	76.4	67.9
16	77.6	69.5

Table 3. Isolated (label-based) phoneme classification and continuous phoneme recognition accuracies after phoneme-level MCE.

testing sentences was tested for the HMMs before and after string-level MCE training. The results obtained are shown in Table 2.

3.2.3. Phoneme-level MCE training of HMMs

Phoneme-level MCE was used with 20-40 iterations of Quickprop to train 3-state context-independent HMMs, starting from the same MLE baseline as used above, on the 3696 TIMIT training sentences (using label information to pick out phonetic segments from each utterance), for various numbers of mixture components per state. After MCE training, both isolated (label-based) phoneme classification and continuous phoneme recognition accuracy for the HMMs were evaluated on the 192 core testing sentences. The results obtained are shown in Table 3.

4. SUMMARY AND DISCUSSION

4.1. String-level MCE training of PBMEC on the ATR isolated word datasets

For the tasks examined, string-level MCE training of PBMEC with a simple, uniform topology yielded error rates that were less than 50 percent of the ML-SSS result for corresponding numbers of parameters (though with a much more sophisticated topology). Furthermore, an average performance of 95.2 % accuracy could be achieved for PBMEC with a total of just 52 mixture components, compared to an average of 95.5 % for ML-SSS with 1200 components, more than 20 times the number of parameters in PBMEC. These are tremendous gains in performance and compactness of representation. These striking improvements are greater than has so far been observed in comparisons between MCE trained recognizers and MLE-based recognizers [2, 3, 6]. Clearly, the ATR isolated word database examined here is not a difficult task, in which the phoneme boundaries are probably quite sharp. MCE is extremely effective in this kind of situation.

4.2. String-level and phoneme-level MCE training of HMMs on TIMIT

String-level MCE yielded significant benefits compared to the MLE baseline for continuous phoneme recognition using the TIMIT database. The difference between MLE and MCE is greatest when using a small number of mixture components.

The results reported here for string-level MCE are very similar to the results obtained for MMI training of an HMM reported in [13]. Though MCE and MMI have different theoretical motivations, they share several features and may result in very similar optimization. In particular, for a difficult task like TIMIT, string-level MCE will be very similar to string-level MMI. Training tokens are nearly always incorrectly recognized, resulting in a misclassification measure that is always positive. In this scenario, illustrated in Figure 1, use of a typical asymmetric MCE loss function, such as the piece-wise linear loss, means that the MCE loss will effectively behave as a linear loss function. Assuming a linear MCE loss function, MCE is very similar to MMI [14].

We found that phoneme-level MCE was more effective than string-level MCE on this task. The results for phoneme-level MCE are as good or better as the MLE results for HMMs with twice the number of mixture components. This is an impressive gain in system compactness. The greater focus in phoneme-level MCE compared to string-level MCE on separating correct from incorrect phonemes may be the reason for the better generalization to unseen data. The results reported here are among the highest reported to date for context-independent HMMs evaluated on the TIMIT database.

REFERENCES

[1] Katagiri, S., Lee, C.-H. and Juang, B.-H. (1991). *New Discriminative Training Algorithms Based on the Generalized Descent Method.* Proceedings of the 1991 IEEE Workshop on Neural Networks for Signal Processing, August 1991, pp 1-10.

[2] McDermott, E. & Katagiri, S. (1992). *Prototype-Based Discriminative Training for Various Speech Units.* Proceedings of the IEEE, ICASSP-92. pp. I:417-420.

[3] Chou, W., Juang, B.-H. and Lee, C.-H. (1992). *Segmental GPD Training of HMM Based Speech Recognizer.* Proceedings of the IEEE, ICASSP-92, pp. I:473-476.

[4] Singer, H. and Ostendorf, M. (1996). *Maximum Likelihood Successive State Splitting.* Proceedings of the IEEE, ICASSP-96. pp. II:601-604.

[5] Chang, P.-C. and Juang, B.-H. (1993). *Discriminative Training of Dynamic Programming Based Speech Recognizers.* IEEE Transactions on Speech and Audio Processing, Vol. 1, No. 2, April 1993, pp. 135-143.

[6] Chou, W., Juang, B.-H. and Lee, C.-H. (1993). *Minimum Error Rate Training Based on the N-Best String Models.* Proceedings of the IEEE, ICASSP-93, pp. II-652-655.

[7] McDermott, E. and Katagiri, S. (1993). *Prototype Based MCE/GPD Training for Word Spotting and Connected Word Recognition.* Proceedings of the IEEE, ICASSP-93. pp. II:291-294.

[8] Robinson, A.J. (1994). *An Application of Recurrent Nets to Phone Probability Estimation.* IEEE Transactions on Neural Networks, Vol. 5, No. 2, pp. 298-305, March 1994.

[9] Lamel, L.F. & Gauvain, J.L. (1993). *High Performance Speaker-Independent Phone Recognition Using CDHMM.* Proceedings of Eurospeech, Vol. 1, pp. 121-124.

[10] McDermott, E. & Katagiri, S. (1997). *Discriminative Training for Speech Recognition.* PhD dissertation, Waseda University, March 1997.

[11] Fahlman, S.E. (1988). *An Empirical Study of Learning Speech in Back-Propagation Networks.* Technical Report CMU-CS-88-162, Carnegie Mellon University, September 1988.

[12] Battiti, R.. (1992). *First- and Second- Order Methods for Learning: Between Steepest Descent and Newton's Method.* Neural Computation, Vol. 4, pp. 141-166.

[13] Valchev, V. (1995). *Discriminative Methods in HMM-based Speech Recognition.* D. Phil dissertation, University of Cambridge, March 1995.

[14] Katagiri, S. and McDermott, E. (1996). *Discriminative Training - recent progress in speech recognition.* In Handbook of Pattern Recognition and Computer Vision (Second edition), Chen, C.H., Pau, L.F., and Wang (eds.), P.S.P. World Scientific Publishing Company.