Now, considering the equal-length segmentation of SPM, 
\[ B(O|\lambda_m) \] can be further expanded as follows:

\[
B(O|\lambda_m) = -0.5T \sum_{d=1}^{D} \frac{\mu_{id}^{(m)} \sigma^{2}_{id}}{\sigma^{2}_{id}} \sum_{t=1}^{T/N} \alpha_{td}^{2} - 2 \mu_{id}^{(m)} \sum_{t=1}^{T/N} \alpha_{td} \sum_{d=1}^{D} \frac{\sigma^{2}_{id}}{\sigma^{2}_{id}} \sum_{t=1}^{T/N} \alpha_{td} \]

where the on-line computational part \( D(O|\lambda_m) \) can be further expressed as

\[
D(O|\lambda_m) = -0.5 \sum_{d=1}^{D} \sum_{i=1}^{N} \left[ O_{id}^{(2)} - 2 \mu_{id}^{(m)} O_{id}^{(1)} \right] \sigma^{2}_{id}
\]

where

\[
O_{id}^{(2)} = \sum_{t=1}^{T/N} \alpha_{td}^{2}
\]

and

\[
O_{id}^{(1)} = \sum_{t=1}^{T/N} \alpha_{td}
\]

It is easy to see that there are \( N \times D \) subtractions, \( N \times D \) multiplications \( [2 \mu_{id}^{(m)} \sigma^{2}_{id}] \) was calculated and stored in advance, \( N \times D \) divisions, and \( N \times D - 1 \) additions to be calculated in \( D(O|\lambda_m) \) for a feature sequence \( O \) with \( T \) frames of \( D \)-dimensional feature vectors, and this computation needs to be repeated by \( L \) times for recognizing an unknown utterance. In addition, there are extra \( T/N \) multiplications and \( (T/N-1) \) additions for evaluating \( O_{id}^{(2)} \), and another \( (T/N-1) \) additions for \( O_{id}^{(1)} \), but they depend only on state index \( i \) and dimension index \( d \), not on the model index \( m \). This reformulation, referred to as the fast-SPM in this paper, can be interpreted as follows. The plain-SPM is calculating a kernel function (referred to as the Gaussian–Euclidean distance function \( b(\alpha; \mu, \Sigma) \) in this work) “frame by frame” as the frame index \( t \) runs from one to \( T \), but the fast-SPM is calculating a kernel function (referred to as the pseudo-Gaussian–Euclidean distance function \( b(i; \mu, \Sigma) \) in this work) “segment by segment” as the segment index \( i \) runs from one to \( N \). Since the computational load of the kernel function for both cases are the same, the computational load of the plain-SPM is about \( T/N \) times of that of the fast-SPM.

The detailed comparison of the computational complexity between the plain-SPM and the fast-SPM are listed in Table VI, with numerical estimations obtained by \( L = 416 \), \( N = 3 \), \( D = 14 \), and \( T = 53 \). We can see from the table that the fast-SPM takes only about \( 1/17 \) of computational time compared to the plain-SPM, which is in turn around \( 1/45 \) compared to the conventional CHMM.

**References**


practical speech recognition, parameters are trained based on a set of speech called the training data while recognition performance is evaluated on a different set called the test data. Usually, speakers whose utterances constitute training and test data are different in each data set, and the relative frequency of each word in the training set is different from that in the test set. Because of such mismatches between the training and test data, it has been frequently observed that a discriminative training technique does not work well for the test data even though high performance improvement is achieved on the training data.

In this correspondence, we propose a robust strategy for maximum mutual information based (MMI-based) hidden Markov model (HMM) parameter estimation [1]–[5]. Since conventional MMI training often adapts the models excessively to the given training data, derived parameters may lose their discriminating capability even when the test data are slightly different from the training data. To avoid this overfitting, some information about the test data is required. But, the distributional characteristics of the test data are completely unknown during the training phase. All that one can do is to predict them based on the current training data. For this reason, we adopt the cross-validation approach, in which parameters are estimated on a part of the data and the recognition performance is evaluated on the other part. We modify the original MMI training algorithm such that the effects of cross-validation can be reflected in the parameter updating rule. By the proposed method, the MMI-based search for HMM parameters can proceed utilizing more realistic recognition errors while maintaining its fast convergence. We also show that the proposed approach can be applied to estimating other types of parameters, such as the interpolating weights [10] and codebook exponents [11], [4]. Moreover, the proposed strategy can easily be extended to discriminative training approaches other than MMI [6]–[8].

This correspondence is organized as follows. In the following section, we propose a method that modifies the conventional MMI-based parameter training algorithm. In Section III, we evaluate its performance with speaker-independent continuous speech recognition experiments. Finally, Section IV presents some concluding remarks.

II. DELETED STRATEGY FOR MMI-BASED PARAMETER TRAINING

In this section, we propose a new approach for MMI-based HMM training. Let θ be a set of all model parameters to be estimated. Then, the MMI criterion can be described as follows:

\[
\hat{\theta}_{\text{MMI}} = \arg \max_{\theta} R(\theta)
\]

where \( R(\theta) \) is the MMI score defined in (1) and (2) as follows:

\[
\hat{\theta}_{\text{MMI}} = \arg \max_{\theta} \prod_{w \in U} \frac{P_{\theta}(w|M_w)P(M_w)}{P_w(w)}
\]

where \( w \) is an acoustic observation sequence in \( U \) (the set of all training data) and \( M_w \) is the corresponding model.

As mentioned earlier, MMI-based parameter estimation performs poorly on the test data while remarkable improvement in recognition accuracy is achieved on the training data. To make it clearer, we demonstrate an example in which two-category classification experiments are executed based on computer-generated samples.

**Example 1:** In this example, we are concerned with classifying samples generated from two different Gaussian sources. For simplicity, we assume that the variances of the two sources are one. Let \( A \) and \( B \) be two categories. Then, the outcomes in each category are distributed as

\[
X \sim \mathcal{N}(m_A, 1), \text{ if } X \in A
\]

and

\[
X \sim \mathcal{N}(m_B, 1), \text{ if } X \in B
\]

where \( X \) represents an outcome and \( \mathcal{N}(\alpha, b) \) is a Gaussian distribution with mean \( \alpha \) and variance \( b \). The recognizer is constructed by two parameters \( \hat{m}_A \) and \( \hat{m}_B \), which classify a sample \( y \) such that

\[
y \in A, \text{ if } (y - \hat{m}_A)^2 < (y - \hat{m}_B)^2
\]

\[
y \in B, \text{ if } (y - \hat{m}_A)^2 > (y - \hat{m}_B)^2
\]

where a tie can be broken in an arbitrary way. Further, let \( \Omega_{\text{train}} \) be a data space in which the \( a \) priori probabilities for \( A \) and \( B \) are \( p \) and \( 1 - p \), respectively. Similarly, assume that in another space \( \Omega_{\text{test}} \), proportions of \( A \) and \( B \) are \( q \) and \( 1 - q \), respectively.

In our experiment with computer generated data, we take \( m_A = -0.5 \) and \( m_B = 0.5 \). 50 samples are drawn from \( \Omega_{\text{train}} \) in order to estimate \( \hat{m}_A \) and \( \hat{m}_B \). After the parameter estimation, classification errors are evaluated on 1000 samples drawn from \( \Omega_{\text{test}} \). We execute such trial 1000 times and average the error rate of each trial. The parameters, \( \hat{m}_A \) and \( \hat{m}_B \) are estimated based on the ML and MMI criteria. ML-based parameter estimation can be done straightforwardly by simply taking the mean of training samples for each category. For MMI-based parameter estimation, we use the approach proposed in [5].

In Table I, experimental results are shown with varying \( p \) and \( q \). The top and middle elements for each \( p \) and \( q \) pair represent the classification error rates of the ML and MMI methods, respectively, and the bottom element will be explained later in this section. One obvious observation is that the performance of MMI-based parameters is more sensitive to the variation of mixture proportions compared to that of the ML-based parameters. Also, it is noticeable that when the same mixture proportions are used both for \( \Omega_{\text{train}} \) and \( \Omega_{\text{test}} \), MMI-based parameters perform better than the ML-based ones except for when \( p = q = 0.5 \). This is due to the incorrect model assumptions lying in the classification rule in (4), which is based on the assumption that the two categories are equally probable. According to these observations, we can conclude that even though the MMI-based parameters perform better than the ML-based ones when incorrect
model assumption is made, their performance is seriously affected by the difference between the training and test data.

Our approach to robust MMI training is motivated by the experimental results shown in the above example. In an iteration of the conventional MMI training algorithm, current parameter points move to new points based on the gradient of the objective function, \( \log R(\theta) \), with respect to the parameters. Hence, MMI-based parameter updating can be considered to comprise of two components, i.e., current parameter values and the gradient of \( \log R(\theta) \). Since the gradient of \( \log R(\theta) \) is determined by both the training data and the current parameter values, we can consider the current parameter values as \textit{a priori} knowledge for the model parameters and the gradient as the new information extracted from the training data using the current parameter values. If we let \( U \) be a set of training data, MMI-based parameter updating can be expressed as

\[
\theta_{\text{move}} = T(\theta, S(U, \theta))
\]

where \( S(U, \theta) \) represents the information extracted from \( U \) based on \( \theta \) and \( T \) is an updating rule such as the one in [9] or other gradient ascent type rule. Moreover, \( S(U, \theta) \) in (5) practically means the gradient of \( \log R(\theta) \) obtained over \( U \) with respect to current parameter values, \( \theta \). The main concept of the newly proposed approach is to use the cross-validation strategy when updating parameter values.

First, we divide \( U \) (the set of all training data) into \( N \) blocks, \( U_1, U_2, \cdots, U_N \). Then, we compute the gradient of \( \log R(\theta) \) while using different parameter values for each block. After computing gradients over all blocks, we update the current parameter values depending on the summation of the gradients. In parallel with this updating, we also update the parameter values used for computing gradient in each block based on the gradients derived in the remaining blocks. We call this approach the deleted MMI training method.

In this method, current value of a parameter is different from that used for computing partial derivative in each block. For that reason, we differentiate a conventional HMM parameter from the ones used for deriving partial derivatives in each block by calling the former as a global parameter and the latter as local parameters. Let \( \theta \) be a set of global parameters and \( \theta^{(i)} \) be the corresponding set of local parameters in \( U_i \). Then, details of the deleted MMI training method are described as follows.

1) Initializing global and local parameter values: Initially, the value for \( \theta \) is set by the corresponding estimate based on \( U \) and the value of \( \theta^{(i)} \) is taken as the estimate for \( \theta \) based on all blocks except for \( U_i \).

2) Computing gradients: We compute \( S(U_i, \theta^{(i)}) \) for \( i = 1, 2, \cdots, N \).

3) Global parameter updating: Global parameter values are updated by

\[
\theta_{\text{move}} = T(\theta, \sum_{i=1}^{N} S(U_i, \theta^{(i)}))
\]

4) Local parameter updating: In parallel with the global parameter updating, each local parameter is also updated by

\[
\theta_{\text{move}}^{(i)} = T(\theta, \sum_{j=1, j \neq i}^{N} S(U_j, \theta^{(j)}))
\]

5) Steps 2)–4) are iterated until convergence condition is met.

Since, at each iteration, \( \theta^{(i)} \) is updated without using the gradient obtained in \( U_i \), the deleted MMI training method can be considered to adopt the cross-validation approach. But, it should be noted that, as shown in Step 4 of the deleted MMI training method, local parameter values are updated not from the current points but from the corresponding global parameter points. This local parameter updating strategy is chosen for the purpose of performing cross-validation while updating the \textit{a priori} knowledge about the model parameters. The convergence condition of the deleted MMI training method is checked according to the value of the deleted mutual information computed at each iteration, which is defined as follows:

\[
\bar{R}(\theta) = \prod_{i=1}^{N} \prod_{w \in U_i} \frac{P_{\theta^{(i)}}(w|M_w)P(M_w)}{P_{\theta}(w)}
\]

where \( \theta^{(i)} \) denotes the set of local parameters used in \( U_i \). Therefore, global parameter updating in deleted MMI training is indirectly evaluated via the local parameter values in each block. Although the deleted MMI training method is not theoretically guaranteed to converge, in most experiments, it has shown convergence similar to that observed in conventional MMI training. In addition, this cross-validatory parameter updating can easily be extended to other discriminative training approaches such as corrective training [6], [7] or generalized probabilistic descent methods [8].

In order to evaluate the performance of the deleted MMI approach, it is first applied to the artificial samples generated in Example 1. For each trial, samples drawn from \( \Omega_{\text{train}} \) are divided into two blocks and then the deleted MMI training method is used to estimate \( \hat{\theta}_{\text{train}} \) and \( \hat{\theta}_{\text{test}} \). The bottom element corresponding to each \( p \) and \( q \) pair in Table I shows the classification error rate yielded by the parameters estimated by the deleted MMI training method. For a fixed value of \( p \), the classification error rate of deleted MMI-based parameters varies similar to that obtained by the conventional MMI-based parameters. But, the range of error rate variation is smaller compared to the case of conventional MMI training. Moreover, error rates for the conventional and deleted MMI training methods have little difference when the mixture proportions are the same both in the training and test data sets. These results show that the deleted MMI-based parameters, compared with the conventional MMI-based ones, are more robust against the mismatch between the training and test data in mixture proportion.

The cross-validation approach can also be applied to estimate other parameters, such as the interpolating weights [10] or codebook exponents [4], [11] based on the MMI criterion. Such parameters are usually used for controlling contributions of several estimates of an HMM parameter in determining probability. For that reason, we call this kind of parameters \textit{control parameters}. When estimating control parameters based on the MMI criterion, we can follow the approach similar to the deleted interpolation technique used for finding robust interpolating weights according to the ML criterion. Let \( \lambda \) be a set of control parameters to be estimated and \( \theta \) be a set of HMM parameters. To apply the cross-validation approach to the estimation of control parameters, we suggest to replace the criterion of (2) by

\[
\hat{\lambda}_{\text{MMI}} = \arg \max_{\lambda} \prod_{i=1}^{N} \prod_{w \in U_i} \frac{P_{\theta^{(i)}}(w|M_w)P(M_w)}{P_{\theta}(w)}
\]

where \( \Gamma^{(i)} = (\lambda, \theta = \theta^{(i)}) \) in which \( \theta^{(i)} \) represents the estimate for HMM parameters based on all blocks except for \( U_i \). Then, \( \hat{\lambda} \) can be updated by using the conventional MMI training algorithm.

III. SPEAKER-INDEPENDENT CONTINUOUS SPEECH RECOGNITION

In this section, the performances of the deleted MMI training method are evaluated with speaker-independent continuous speech recognition experiments.
A. Baseline Recognition System

The vocabulary consists of 102 Korean words representing month, day, date and time. Ninety speakers (43 males and 47 females) uttered 20 to 30 sentences to construct the data base used for training and evaluation. We used 12th-order linear predictive coding (LPC) cepstral coefficients and differenced LPC cepstral coefficients as the feature vectors, and extracted them in every frame of 10 ms. Two separate codebooks were constructed such that the number of codewords is 256 for each codebook. Twenty-seven phoneme models involving silence model were used as the basic units of recognition. Each unit was modeled by a three-state discrete HMM which is a simple left-to-right model without skipping. All the HMM parameters were trained according to the ML criterion using the segmental approach.

B. Experimental Results with Deleted MMI Training

Three different sets of training and test data were taken in order to make specific mismatch conditions. The first set represents the mismatch in speaker distribution. Utterances from 70 speakers (33 males and 37 females) constructed the training data in which there were 1631 sentences and 5119 words, and those from the other 20 speakers were used to form the test data containing 439 sentences and 1448 words. The second set was constructed such that the distribution of words could be far different from training to test data. For this, we applied the binary tree algorithm [12] to the total data set and obtained a set of training and test data, which were separated far from each other in word distribution. The training and test data were consisted of 3865 and 2702 words, respectively. The last set represents a mismatch condition similar to the second set. But, instead of using the distribution of words, the distribution of phonemes was used for dividing the training and test data. As a result, we obtained training data of 3363 words and test data of 3204 words. These three sets are represented by $DB_{\text{speaker}}$, $DB_{\text{word}}$ and $DB_{\text{phoneme}}$, respectively.

First, we applied the conventional MMI training algorithm to estimate the HMM parameters. For fast convergence, we used the $N$-best algorithm [3] in the sentence level and took the most probable three sentence models into consideration. Initial value for each parameter was set by the value obtained through ML training.

Next, we estimated the parameter values based on the deleted MMI training method for which we divided the training data set into two blocks. For $DB_{\text{speaker}}$, the training data were divided in order to construct separate speaker groups for each block. Thus, pronunciations of a speaker were wholly assigned to one of the two blocks. As for $DB_{\text{word}}$ and $DB_{\text{phoneme}}$, we applied the binary tree algorithm to split the training data such that the two blocks were separated far from each other in word and phoneme distributions, respectively. We could observe that the convergence behaviors of deleted MMI with $DB_{\text{speaker}}$, $DB_{\text{word}}$ and $DB_{\text{phoneme}}$ were nearly the same to those of conventional MMI. That is, error rate in the training data set continuously decreased until it reached the lowest error rate, and after then it fluctuated.

Recognition error rates yielded by the conventional and deleted MMI methods are shown in Table II, where the results for ML training are also given for the purpose of comparison. For all the data sets, the recognition results of deleted MMI show enhanced performance compared to those of conventional MMI. However, the improvements in recognition rate are little. One major reason for this is that the recognition error patterns observed in each block during the training phase are similar and as a result, the gradients of the objective function derived from respective blocks become similar.

IV. CONCLUSIONS

In this correspondence, the deleted MMI training method was proposed as a robust approach to discriminative parameter estimation. Even though a discriminative training algorithm may be robust to incorrect modeling assumption, it will be useless if statistical characteristics of the speech data for evaluating the recognition performance are different from those in the training data. We modified the conventional MMI training algorithm such that the effects of cross validation could be reflected in the parameter updating rule. By the proposed approach, the MMI-based search for HMM parameters can proceed utilizing more realistic error patterns while maintaining its convergence behavior. How to divide the training data according to the mismatch condition will be one of the remaining future study areas.

REFERENCES

A Comparison of Constrained Trajectory Segment Models for Large Vocabulary Speech Recognition

Ashvin Kannan and Mari Ostendorf

Abstract—This paper compares parametric and nonparametric constrained-mean trajectory segment models for large vocabulary speech recognition, extending distribution clustering techniques to handle polynomial mean trajectory models for robust parameter estimation. The parametric model has fewer free parameters and gives similar recognition performance to the nonparametric model, but has higher recognition costs.

Index Terms—Distribution clustering, polynomial mean trajectory, segment model.

I. INTRODUCTION

Most state-of-the-art speech recognition systems use hidden Markov models (HMM’s) as acoustic models, typically with output distributions that are mixtures of Gaussians with diagonal covariances. This coupled with an independent-frame assumption, allows spectral trajectories in time within a state to be discontinuous. However, we know that the smooth motion of the articulators imposes natural constraints on the trajectory. Attempts to model trajectory constraints are made in segment models, which can be considered as a generalization of HMM’s where each “state” (which may represent a phonetic or subphonetic unit) emits a random length sequence of observations [1]. Such constrained trajectory models can be classified into two categories: nonparametric models, e.g., [2] and [3], and parametric models, e.g., [4] and [5].

In this work, we compare two constrained trajectory segment models: a nonparametric model, the stochastic segment model (SSM) [3], with a parametric model, the polynomial-trajectory segment model (PSM) [4], in the context of large vocabulary continuous speech recognition (LVCSR) with the Wall Street Journal [6] and Switchboard corpora [7]. Good performance on LVCSR tasks is typically achieved by modeling co-articulation effects (phonetic context) through clustering context-dependent models. Such a framework has been developed for the SSM with resulting performance comparable to HMM’s [8], but none exists for the PSM. So far, PSM’s have been used only in constrained tasks like vowel-classification, secondary processing for word-spotting [4] and isolated word recognition [5], where they have been shown to be better than conventional HMM’s. In this work, we extend the PSM framework and compare it to the SSM, which for similar numbers of distributions (SSM regions versus HMM states) is comparable to an HMM [9].

For constrained trajectory segment models, in general, the probability of a variable-length sequence of \( d \)-dimensional vector observations \( Y = [y_1, \ldots, y_n] \) is described by

\[
p(Y | \{\mu_i\}, \{\Sigma_i\}) = p(Y | \{\mu_i\}, \{\sigma_i\}) = P(n)p(Y | \{\mu_i\}, \{\Sigma_i\}, n)
\]

where \( P(n) \) is the duration probability and the observations are assumed to be conditionally independent given the segment length and a deterministic mapping \( \gamma(j) \) of the time sequence to the distribution sequence. The mapping constrains the mean trajectory, hence the term “constrained trajectory.” The SSM and the PSM differ primarily in the representation of the sequence of distributions for a segment, or equivalently the sequence of Gaussian distribution parameters \( \{\mu_i\}, \{\Sigma_i\} \), as described below.

In the SSM, a phone is associated with a finite number of regions, each represented by a full covariance Gaussian. In this case, \( \gamma(j) \) is an index to the region chosen for the \( j \)-th frame, represented by the \( d \)-dimensional Gaussian \( N(\mu_{\gamma(j)}, \Sigma_{\gamma(j)}) \). Observations within a region are assumed to have been generated from the same distribution, i.e., for unimodal distributions, the trajectory of the means in feature-space is constrained to be piecewise constant since \( \mu_{\gamma(j)} = \mu_k \) for all \( j \) such that \( \gamma(j) = k \).

For the PSM, observations from a segment are assumed to be generated by a Gaussian process whose mean is characterized by a polynomial in time in \( d \)-dimensional vector space, e.g., [4], [5]. In this work, we will assume that the covariance of the process is constant over a PSM segment. For an \( r - 1 \) order polynomial, the time-varying mean trajectory can be written as \( \mu(t) = b_1 + b_2 t + \ldots + b_r t^{r-1} \) for \( t \in [0, 1] \), where \( t \) represents normalized time as in [4]. The specific means used to evaluate an \( n \)-length segment of frame-level observations \([y_1, \ldots, y_n]\) are found by sampling the trajectory at \( n \) points described by \([\mu_1, \ldots, \mu_n]\) = \( B Z^n \), where \( B = [b_1, \ldots, b_r] \) is a \( d \times r \) matrix of polynomial coefficients and \( Z^n \) is an \( n \times r \) time-sampling matrix. If the observed segment is a linearly sampled realization of the complete trajectory, \( Z^n \) has the following form:

\[
Z^n = [z_1, \ldots, z_n]
\]

where \( z_j = [t - 1 \over n - 1]^2 \ldots (j - 1 \over n - 1)^{(r - 1) \over 2} \) (2)

where the superscript \( T \) denotes vector transpose. (For simplicity and direct comparison of the parametric and nonparametric approaches, the time sampling is linear in both cases.) A related model is that of [5], which is different in that time is not normalized, but a “sampling” matrix can be trivially defined to cast it in our formalism.

In this work, a phone segment is modeled with two PSM’s, with each PSM describing observations for one region (so we have two