EFFICIENT ML TRAINING OF CDHMM PARAMETERS BASED ON PRIOR EVOLUTION, POSTERIOR INTERVENTION AND FEEDBACK

Qiang Huo, Nathan Smith and Bin Ma

Department of Computer Science and Information Systems,
The University of Hong Kong, Pokfulam Road, Hong Kong (e-mail: qhuo@csis.hku.hk)

ABSTRACT

We present an efficient maximum likelihood (ML) training procedure for Gaussian mixture continuous density hidden Markov model (CDHMM) parameters. This procedure is proposed using the concept of approximate prior evolution, posterior intervention and feedback (PEPIF). In a series of experiments for training CDHMMs for a continuous Mandarin Chinese speech recognition task, the new PEPIF procedure achieves a 4-fold speed-up in terms of user CPU time over that of the Baum-Welch algorithm in producing models of given likelihood or recognition accuracy.

1. INTRODUCTION

Currently, Gaussian mixture CDHMMs remain predominant for basic speech unit modeling. The most popular HMM parameter estimation method from a finite amount of training data is still the batch-mode ML training known as the Baum-Welch algorithm (or its decision-directed variation known as segmental ML or Viterbi training). Over the years, speech researchers have observed that in order to obtain a good recognition performance, instead of aiming at a very accurate estimation of the model parameters, obtaining a good rough estimate seems to be more efficient; and it has proved to work well in practice. It was also observed that although the Baum-Welch algorithm eventually has a linear speed of convergence, the first several iterations usually achieve the most significant likelihood increase. Speech practitioners take advantage of this property and in most cases just run several Baum-Welch iterations (say within 10) in HMM training. Actually, this has already been a heavy burden. The fact that researchers have been able to afford the use of an increasing amount of training data for HMM training in the past two decades is simply because of the availability of increasingly powerful machines in terms of speed, memory, and hard disk. When the increasing amount of training data becomes available, there is a real need to develop a more efficient ML training method so as to make *speech engineering* possible even in small research groups with limited computational resources.

A study with an explicit goal of speeding up the ML training for CDHMM has been reported in [1]. One of the algorithms in [1] is a special case of the Quasi-Bayes (QB) learning framework proposed in [2]. As we discussed in [2], QB learning can also be used as a Bayesian tool to achieve efficient ML training by using the concept of *approximate prior evolution, posterior intervention and feedback* (PEPIF). Gotoh *et al.* [1] did not realize the underlying theoretical mechanism of the approximate prior evolution and failed to provide a *posterior intervention* mechanism before feeding back the posterior distribution as a prior for the next iteration. As we will show in this study, such a *forgetting mechanism* is important to prevent a possible premature convergence of the algorithm. We will describe how to use the PEPIF algorithm to perform an efficient ML training, and will compare it quantitatively to the Baum-Welch algorithm and the algorithm in [1].

2. METHODOLOGY

Consider a collection of M CDHMM's $\Lambda = \{\lambda_q\}_{q=1,\cdots,M}$, where $\lambda_q = (\pi^{(q)}, A^{(q)}, \theta^{(q)})$ denotes the set of parameters of the q-th N-state CDHMM used to characterize the q-th speech unit. In this compact notation, $\pi^{(q)}$ is the initial state distribution, $A^{(q)} = [a_{ij}^{(q)}]$ is the transition probability matrix, and $\theta^{(q)}$ is the parameter vector composed of mixture parameters $\theta_i^{(q)} = \{\omega_{ik}^{(q)}, m_{ik}^{(q)}, r_{ik}^{(q)}\}_{k=1,\cdots,K}$ for each state i. The state observation probability density function (pdf) is a mixture of K multivariate Gaussian pdf's: $p(\mathbf{x}|\theta_i^{(q)}) = \sum_{k=1}^{K} \omega_{ik}^{(q)} \mathcal{N}(\mathbf{x}|m_{ik}^{(q)}, r_{ik}^{(q)})$, where $\omega_{ik}^{(q)}$ is the mixture coefficient, $m_{ik}^{(q)}$ is the D-dimensional mean vector, and $r_{ik}^{(q)}$ is the $D \times D$ precision (inverse covariance) matrix, with its d-th diagonal element being $\sigma_{ikd}^{-2(q)}$. In the following discussion, we will drop the notation regarding the speech unit index q. Furthermore, let $\mathcal{X}$ be a set of training samples. Our problem is then defined on how to *efficiently* obtain an ML estimate of Λ from $\mathcal{X}$.

We use the Bayesian formulation as a *tool* to derive such an efficient ML training algorithm. We consider the uncertainty of the HMM parameters Λ by treating them as if they were random variables. Given a seed model (initial estimate) Λ_{seed}, we can summarize our prior knowledge about Λ as an initial prior pdf $p(\Lambda|\varphi^{(0)})$ with *hyperparameters* $\varphi^{(0)}$. If we *randomize* training samples in $\mathcal{X}$ and divide them into B blocks, then we can artificially create a training set $\mathcal{X}_1^B = \{\mathcal{X}_1, \mathcal{X}_2, \cdots, \mathcal{X}_B\}$, where $\mathcal{X}_i$'s can be viewed as B independent, incrementally obtained sets of observation samples. With $p(\Lambda|\varphi^{(0)})$ and $\mathcal{X}_1^B$, we can obtain an approximate posterior pdf $p(\Lambda|\mathcal{X}_1^B)$ by using the *prior evolution* method described in [2]. Taking a point estimate Λ_1 (e.g. mode) from $p(\Lambda|\mathcal{X}_1^B)$ will give us HMM parameters updated after one pass of $\mathcal{X}$. The training procedure can

0-7803-6293-4/00/$10.00 ©2000 IEEE.

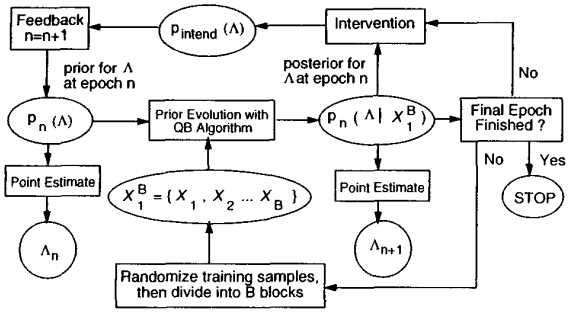

Figure 1: Block Diagram of PEPIF Procedure.

either terminate at this point or can continue. If the latter is the case, then we need to *flatten* the $p(\Lambda|\mathcal{X}_1^B)$ by using an *intervention scheme* to obtain a new pdf $p_{intend}(\Lambda)$. We then *feedback* this pdf to serve as the prior for the next pass of $\mathcal{X}$. At the same time, we also need to randomize $\mathcal{X}$ again and re-divide it into B blocks to obtain a new $\mathcal{X}_1^B$. This completes one *epoch* of our training procedure. The procedure will continue by repeating the above steps as schematically shown in Fig. 1. In the following discussion, we will refer to this new procedure as **PEPIF**.

In this study, we only consider the case of CDHMMs with diagonal precision matrices. All of the prior and posterior pdf's are assumed to be a product of Dirichlet pdf's and normal-gamma pdf's with the set of hyperparameters $\{\eta_i\}$, $\{\eta_{ij}\}$, $\{\nu_{ik}\}$, $\{\tau_{ik}\}$, $\{\alpha_{ik}\}$, $\{\beta_{ikd}\}$. Readers are referred to [2] for a detailed explanation of these distributions as well as the approximate prior evolution method based on a Quasi-Bayes approach. In the following section, we will describe in detail 1) how to derive the initial prior pdf $p(\Lambda|\varphi^{(0)})$ from the seed model Λ_{seed}; and 2) the intervention scheme we used in the experiments reported in this study.

3. IMPLEMENTATION ISSUES

3.1. From Seed Model to Initial Prior Distribution

There are currently two different initialization methods which are termed τ-initialization and prior-weight initialization.

3.1.1. τ-initialization

τ-initialization requires a user-defined seed model Λ_{seed}, together with a value of τ which is used to control the broadness of the initial prior pdf $p(\Lambda|\varphi^{(0)})$. The bigger the value of τ, the sharper the distribution $p(\Lambda|\varphi^{(0)})$. Using these, the hyperparameters $\varphi^{(0)}$ [2] are initialized as follows:

$$\eta_i^{(0)} = 1 + \pi_i^{(seed)} \times N \times \tau \quad (1)$$

$$\eta_{ij}^{(0)} = 1 + a_{ij}^{(seed)} \times N \times \tau \quad (2)$$

$$\nu_{ik}^{(0)} = 1 + \omega_{ik}^{(seed)} \times K \times \tau \quad (3)$$

$$\mu_{ik}^{(0)} = m_{ik}^{(seed)} \quad (4)$$

$$\tau_{ik}^{(0)} = \omega_{ik}^{(seed)} \times K \times \tau \quad (5)$$

$$\alpha_{ik}^{(0)} = \frac{1}{2} + \frac{1}{2}\tau_{ik}^{(0)} \quad (6)$$

$$\beta_{ikd}^{(0)} = \frac{1}{2}\tau_{ik}^{(0)}\sigma_{ikd}^{2(seed)} \quad (7)$$

3.1.2. Prior-weight initialization

This initialization requires a seed model which is then input into the Baum-Welch algorithm. The following statistics are collected from a single pass through the training data $\mathcal{X}$:

$$g_{ij}^{(init)} = \sum\nolimits_t \gamma_t(i,j) \quad (8)$$

$$c_{ik}^{(init)} = \sum\nolimits_t \zeta_t(i,k) \quad (9)$$

$\gamma_t(i,j) = \Pr(s_t = i, s_{t+1} = j|\mathcal{X}, \Lambda_{seed})$ is the probability of being in state i at time t and state j at time $t+1$, (given training data $\mathcal{X}$ and seed HMM parameters Λ_{seed}); and $\zeta_t(i,k) = \Pr(s_t = i, l_t = k|\mathcal{X}, \Lambda_{seed})$ is the probability of observing $\mathbf{x}_t$ in mixture component k of state i. Using the above statistics, and a prior weight ϵ_0, the hyperparameters are initialized as follows:

$$\eta_i^{(0)} = 1 + \epsilon_0 \times \sum\nolimits_j g_{ij}^{(init)} \quad (10)$$

$$\eta_{ij}^{(0)} = 1 + \epsilon_0 \times g_{ij}^{(init)} \quad (11)$$

$$\nu_{ik}^{(0)} = 1 + \epsilon_0 \times c_{ik}^{(init)} \quad (12)$$

$$\mu_{ik}^{(0)} = m_{ik}^{(seed)} \quad (13)$$

$$\tau_{ik}^{(0)} = \epsilon_0 \times c_{ik}^{(init)} \quad (14)$$

$$\alpha_{ik}^{(0)} = \frac{1}{2} + \frac{1}{2}\epsilon_0 \times c_{ik}^{(init)} \quad (15)$$

$$\beta_{ikd}^{(0)} = \frac{1}{2}\epsilon_0 \times c_{ik}^{(init)} \times \sigma_{ikd}^{2(seed)} \quad (16)$$

Similarly to the role of τ discussed above, the prior weight ϵ_0 controls the broadness of the prior pdf $p(\Lambda|\varphi^{(0)})$.

3.2. Posterior Intervention Scheme

In posterior-prior feedback, the posterior is used as the prior for the next epoch of prior evolution. Depending on the intended purpose, a quite flexible posterior intervention scheme can be designed: 1) As the posterior is fed into the next stage as the prior, the distribution may be widened. The effect of this is to *forget* some information learned in the previous epoch; 2) In the same way, the distribution may be narrowed to *consolidate* the information already present in the posterior. In QB formulation, this can be achieved by refreshing the hyperparameters of the approximate posterior pdf as follows:

$$\hat{\eta}_i = 1 + \epsilon \times (\eta_i - 1) \quad (17)$$

$$\hat{\eta}_{ij} = 1 + \epsilon \times (\eta_{ij} - 1) \quad (18)$$

$$\hat{\nu}_{ik} = 1 + \epsilon \times (\nu_{ik} - 1) \quad (19)$$

$$\hat{\tau}_{ik} = \epsilon \times \tau_{ik} \quad (20)$$

$$\hat{\mu}_{ik} = \mu_{ik} \quad (21)$$

$$\hat{\alpha}_{ik} = 0.5 + \epsilon \times (\alpha_{ik} - 0.5) \quad (22)$$

$$\hat{\beta}_{ikd} = \epsilon \times \beta_{ikd} \quad (23)$$

where $\epsilon > 0$ is called the *refreshing factor* or *intervention factor*. The *forgetting effect* occurs when $\epsilon < 1.0$ and the *consolidating effect* occurs when $\epsilon > 1.0$. However, if ϵ is taken greater than 1.0 in the early stages of the prior evolution, the algorithm will converge to a result very quickly,

although it may not be a local maximum of the likelihood function. In this study, we are interested in the efficient ML training of CDHMM parameters for *large amounts* of training data. In this case, even if we start from a non-informative prior, after one epoch of prior evolution using the QB algorithm, the resultant approximate posterior pdf will probably become quite sharp. If we directly feedback this posterior pdf as a prior for the next epoch as performed in [1], the algorithm might converge very quickly to a result which is still far away from the local maximum of the likelihood function. Therefore, a natural way of controlling the convergence behavior of the algorithm is to make the refreshing factor *adaptive*: starting from a small refreshing factor, we gradually increase the value of the refreshing factor with successive epochs. By doing this, we should avoid possible premature convergence, while still ensuring the final convergence of the algorithm. We studied several refreshing schedules, and found that the following *exponential refreshing schedule* works quite well:

$$\epsilon_n = \begin{cases} \epsilon_0 \times b^{n-1} & \text{if } \epsilon_n < 1 \\ 1 & \text{otherwise} \end{cases} \quad (24)$$

where ϵ_n is the refreshing factor for Epoch n, ϵ_0 is the initial refreshing factor (to which the prior weight is assigned in the prior-weight initialization scheme), and b is the base of the exponential (this controls how fast the values of ϵ_n are increased).

4. EXPERIMENTS

As we discussed above, the PEPIF procedure requires the setting of some control parameters which include the initialization parameters and the refreshing schedule. Extensive experiments on both artificial data and speech data have been performed 1) to discern the effect of those control parameters on performance (in terms of likelihood increase and speech recognition accuracy) and 2) to check what the best set of control parameters should be. Because of the space limitation, we can only here report part of the results.

4.1. Experimental Setup

Experiments are performed for continuous speech recognition of Putonghua (Mandarin Chinese). The database we used is the HKU96 Putonghua Corpus developed in our lab. The HKU96 corpus consists of a total of 20 native Putonghua speakers, 10 females and 10 males. All speech recording were made in a quite room with a single National Cardioid Dynamic Microphone. Speech was digitized using a Sound Blaster 16 ASP A/D card at 16-bit accuracy and with a sampling rate of 16KHz. The 39-dimensional feature vectors used in this study consist of 12 MFCCs and log-scaled energy normalized by the peak of the individual sentence, plus their first and second order derivatives. Sentence-based cepstral mean subtraction (CMS) is applied for acoustic normalization, both in training and testing.

18224 utterances (about 15.5 hours of raw speech) from 18 speakers (9 females and 9 males) are used for training. A further two speakers (1 female and 1 male) are used for speaker-independent (SI) testing; there are 996 utterances

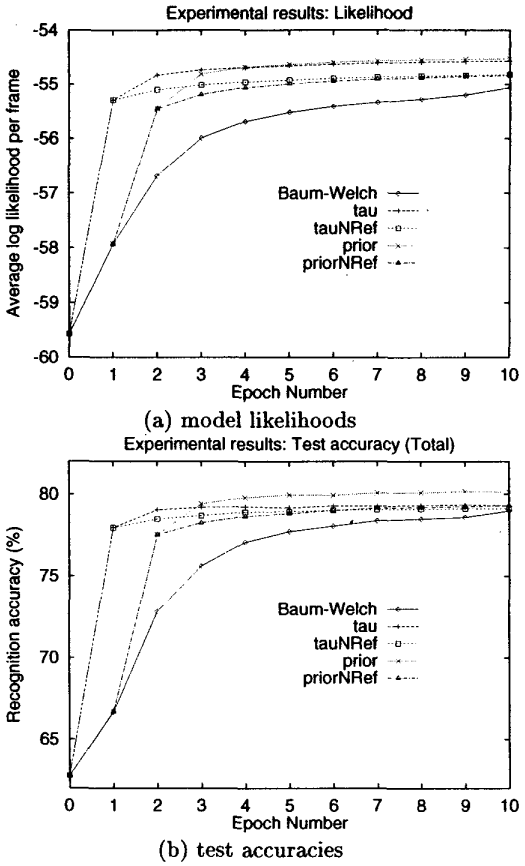

Figure 2: Likelihoods and Test Accuracies of Models trained by several algorithms as a function of Epoch number.

from the female speaker, and 975 utterances from the male speaker. The recognition task is the recognition of 410 Putonghua *base syllables* disregarding tones. The recognition network enforces silence at the start and end of sentences and allows optional silences between syllables. As for the syllable language model, a uniform grammar with a syllable perplexity of 411 (i.e. each syllable can be followed by any of the 410 base syllables and silence) is used. All the recognition experiments are performed using the re-scoring technique provided in the HTKv2.1 toolkit [3].

The system to be trained is an SI, cross-syllable-triphone, decision-tree-based, tied-state system. The adopted context-independent (CI) phone set consists of 36 phones plus silence. With this phone set definition, there are 8022 triphones in Putonghua. Among them, 5595 triphones are observed in the training data set. Each phone is modeled by a left-to-right, three-emitting-state CDHMM without state skipping. Each state has 4 Gaussian mixture components with each component having a diagonal covariance matrix. A special three-emitting-state CDHMM is also used for silence modeling. The initial models are prepared by using the HTKv2.1 toolkit. Using the initial models as seed models, we perform ML training both with the Baum-Welch algorithm as implemented in HTKv2.1, and with the PEPIF procedure under different settings of control parameters.

Table 1: The User CPU Time Overhead for a Single Epoch of PEPIF with Different Block-Sizes, Compared to the Baum-Welch Algorithm.

Method	Block Size	User CPU Time Overhead (%)	
		No Pruning	Using Pruning
PEPIF	20	11.4	21.0
PEPIF	100	4.9	14.0
PEPIF	1000	2.1	10.4
Baum-Welch	18224	0.0	0.0

4.2. Comparison of Learning Curves

Fig. 2 (a) compares the learning curves (as a function of Epoch number) of several algorithms in terms of the average log likelihood per frame. Fig. 2 (b) shows the corresponding learning curves in terms of the syllable recognition accuracy averaged over 2 testing speakers. The curves labeled 'Baum-Welch' correspond to the results obtained by the Baum-Welch algorithm. Other curves correspond to the results obtained by PEPIF with different experimental setup. The curves labeled 'tau' correspond to the results with the following baseline setup: 1) For initial prior specification, τ-initialization is used with an initial τ-value of 1.0; 2) For prior evolution, sequential QB is used with a block-size of 20 sentences; 3) For posterior intervention, the exponential refreshing schedule is used with initial refreshing factor ϵ_0 set to 0.01 and base b set to 1.5. The curves labeled 'prior' correspond to the results with the above baseline setup excepting that prior-weight initialization is used for initial prior specification. These two sets of experiments are designed to compare the efficacy of the two initial prior specification methods discussed in Section 3.1. In order to examine the effectiveness of the posterior intervention scheme discussed in Section 3.2, another two sets of experiments are performed that start from the above two different priors. This time, a refreshing factor of 1.0 (i.e. no refreshing) is always used for each epoch. The learning curves are labeled as 'tauNRef' and 'priorNRef' respectively.

Several observations can be made from these results: 1) The PEPIF increases the model likelihoods much faster than does Baum-Welch (especially in the early epochs), and attains a better likelihood after 10 epochs. These results are also translated into test accuracies, showing that the higher-likelihood models produced by PEPIF are actually 'better' models; 2) The use of a refreshing schedule has a beneficial effect in improving model likelihoods when compared to a 'no-refreshing' schedule as done in [1]; 3) Prior-weight initialization appears to be a better initialization method than τ-initialization with respect to final likelihoods and accuracies. However, for the prior-weight initialization method, Epoch 1 in Fig. 2 is essentially a batch-mode Baum-Welch iteration. The likelihood increase in this epoch is thus much smaller than that yielded by using τ-initialization. After 3 epochs, the prior-weight curves surpass the τ-initialization curves. From the above results, we conclude that PEPIF will be very efficient for the ML training of CDHMM parameters when 'early stopping' is used. For example, to compare the performance of baseline PEPIF with Baum-Welch, the likelihood of PEPIF at Epoch 2 is -54.828; this is higher than the likelihood of Baum-Welch at Epoch 10 at -55.064. This indicates a 5-fold speed-up in the numbers of epochs. The accuracy curves show that this speed-up generalizes to unseen data. The syllable recognition accuracy at Epoch 2 for PEPIF is 79.04%, whereas at Epoch 10 for Baum-Welch, it is 79.3%. The speed-up is therefore also maintained in the accuracy curves.

4.3. Comparison of Computational Complexity

However, in comparison with the Baum-Welch algorithm, the PEPIF procedure introduces an overhead for each epoch. This overhead mainly depends on the block size: the smaller the size, the more the number of data blocks, and thus the more updates of parameters are required in one epoch. Table 1 shows the user CPU time overhead for a single epoch of PEPIF with different block-sizes, compared to the Baum-Welch algorithm. If pruning in the Forward-Backward algorithm is virtually disabled [3], we observe a small overhead of 11.4% $\sim$ 2.1% for a block-size of 20 $\sim$ 1000. If the default pruning in HTKv2.1 is used, a bigger overhead of 21.0% $\sim$ 10.4% is observed. However, PEPIF successfully trains models with fewer numbers of epochs than does Baum-Welch. Therefore, PEPIF offers the user a faster technique for obtaining a set of models of a given likelihood or accuracy. For the previous example of early stopping at Epoch 2, using a block-size of 20 and keeping pruning, a 4-fold speed-up in terms of user CPU time is achieved.

5. SUMMARY

The PEPIF algorithm has been compared against the Baum-Welch algorithm, and the effect of varying various control parameters for the PEPIF algorithm has been investigated. The experiments in this study show that generally PEPIF produces a faster increase in likelihood and accuracy than does Baum-Welch. Though PEPIF has a processing overhead, it still offers a speed-up over Baum-Welch in the time taken to produce models of given likelihood or accuracy. PEPIF may also produce models of higher likelihood and accuracy than Baum-Welch can produce even with a large number of training epochs. This shows a good asymptotic convergence property. It is observed that the PEPIF algorithm works reliably in all of the experiments we performed on both artificial data and real speech data. We recommend to the community the routine use of this new algorithm. We will report elsewhere a more theoretical analysis of the PEPIF procedure, and more experimental results on the sensitivity of the algorithm to different settings of the control parameters.

REFERENCES

[1] Y. Gotoh et al., "Efficient training algorithms for HMMs using incremental estimation," *IEEE Trans. on SAP*, Vol. 6, pp.539-548, 1998.

[2] Q. Huo and C.-H. Lee, "On-line adaptive learning of the continuous density hidden Markov model based on approximate recursive Bayes estimate," *IEEE Trans. on SAP*, Vol. 5, pp.161-172, 1997.

[3] S. Young et al., *The HTK Book* (for HTK Version 2.1), Cambridge University, 1997.