MAXIMUM-LIKELIHOOD ADAPTATION OF SEMI-CONTINUOUS HMMS BY LATENT VARIABLE DECOMPOSITION OF STATE DISTRIBUTIONS

Antoine Raux and Rita Singh
School of Computer Science, Carnegie Mellon University, USA
{antoine, rsingh}@cs.cmu.edu

ABSTRACT

Compared to fully-continuous HMMs, semi-continuous HMMs are more compact in size, require less data to train well and result in comparable recognition performance with much faster decoding speeds. Nevertheless, the use of semi-continuous HMMs in large vocabulary speech recognition systems has declined considerably in recent years. A significant factor that has contributed this is that systems that use semi-continuous HMMs cannot be easily adapted to new acoustic (environmental or speaker) conditions. While maximum likelihood (ML) adaptation techniques have been very successful for continuous density HMMs, these have not worked to a usable degree for semi-continuous HMMs. This paper presents a new framework for supervised and unsupervised ML adaptation of semi-continuous HMMs, built upon the paradigm of probabilistic latent semantic analysis. Experiments with a specific implementation developed under this framework demonstrate its effectiveness.

1. INTRODUCTION

In most HMM-based speech recognition systems, the parameters of the HMMs are usually trained from large corpora of training data, recorded from many speakers under diverse recording conditions. As a result, the HMMs represent the *average* characteristics of the sound across all conditions, but do not model any specific speaker or recording condition perfectly. While generalizability across wide range of acoustic conditions is a desirable characteristic for any practical system, it is well known that the recognition performance of systems that use these HMMs can be greatly improved by updating, or *adapting*, the values of the HMM parameters continually to represent the current speaker and recording conditions. Since condition-specific or speaker-specific data are often sparsely available, this is also the more pragmatic approach.

1.1 The problem of adaptation in semi-continuous HMM based systems

In both fully-continuous and semi-continuous HMMs, the state output densities are typically Gaussian mixtures. The difference between the two is that in semi-continuous HMMs, the components of all Gaussian mixture densities are common - different states merely assign different mixture weights to them [1]. In both types of HMMs, the process of adaptation is not simple. There is usually insufficient data from the current speaker or recording condition to update all HMM parameters, and direct parameter re-estimation using only the available data is usually counterproductive. Instead, conventional HMM-based systems use one of two approaches - *maximum a posteriori* (MAP) adaptation and model-based *maximum likelihood* (ML) adaptation. In MAP adaptation, the *a priori* distribution of the optimal HMM parameters for various speaker and noise conditions is learned beforehand during training (this is not to be confused with the distribution of the data itself). When adapting the system to a given speaker, the parameters are updated to jointly maximize the likelihood of the data from the speaker and the likelihood of the parameters on the *a priori* distribution of parameters. The use of the *a priori* distribution ensures that the updated model parameters will continue to generalize to new speech from the current speaker or recording condition. In model-based adaptation, on the other hand, no *a priori* parameter distributions are assumed, and a simple model of how the generic HMM parameters must be transformed to best represent the new data is hypothesized. The model is chosen to have a small number of model parameters that are common to the entire set of HMM parameters. Since the number of model parameters is small, they can be learned from small amounts of adaptation data, and then used to transform the HMM parameters to represent new speech. In practical systems that use fully-continuous systems, MAP adaptation has been found to be unreliable for unsupervised adaptation. Only model-based ML adaptation has been shown to be effective.

Both supervised and unsupervised ML adaptation of HMM parameters has hitherto been largely unsuccessful for semi-continuous HMMs. In semi-continuous HMMs the bulk of the modeling is done by the mixture weights of Gaussians. The vectors of mixture weights for the various tied states in the HMM do not populate the space of these vectors densely. Rather they lie scattered on the surface of the probability simplex. They cannot be clustered by conventional mechanisms. Further, it is difficult to define generalizable transforms with a small number of parameters that will apply to clusters of mixture weight vectors such that the transformed vectors remain on the simplex, while simultaneously fitting to the adaptation data in a generalizable manner. Hence, in the case of semi-continuous HMMS, while MAP adaptation techniques have been proposed [2], model-based maximum likelihood techniques have not been effectively developed.

In the rest of this paper we describe our new approach to adaptation of semi-continuous HMMs by non-negative matrix factorization of mixture weights in a PLSA (Probabilistic Latent Semantic Analysis) framework.

2. MIXTURE WEIGHTS AS SCALED JOINT PROBABILITIES OF TIED STATES AND GAUSSIANS

In state-of-art HMM-based speech recognition systems, a small number of densities (typically a few thousand) are shared by the HMM states for all the sound units modeled by the system. HMMs states that share a common density are referred to as *tied states* or *senones*. In semi-continuous HMMs, all HMM state densities share a mixture of Gaussians from a codebook of Gaussians. The only distinction between the various densities lies in the mixture weights that are assigned to the Gaussians. Senones in semi-continuous HMMs further share the mixture weights.

Let K be the number of Gaussians in any state output density. Let $G(x;\mu_k,\Phi_k)$ represent the k^{th} Gaussian, with mean μ_k and variance Φ_k. The state output density for any senone s has the form:

$$P(x|s) = \sum_{k=1}^{K} w_{s,k} G(x;\mu_k,\Phi_k) \tag{1}$$

where $w_{s,k}$ is the mixture weight that must be applied to the k^{th} Gaussian in constructing the density for s. $w_{s,k}$ represents $P(k|s)$, the *a priori* probability of the k^{th} Gaussian, given the senone s. From Bayes' rule, $w_{s,k}$ can be expressed in terms of $P(k,s)$, the joint probability of senone s and the k^{th} Gaussian, as

$$P(k|s) = C(s)P(k,s) \tag{2}$$

where $C(s)$ is a scaling constant. Let N be the total number of senones. The joint probabilities of senones and Gaussians can be arranged in a $K \times N$ matrix, P as

$$P = \begin{bmatrix} P(1,1) & P(1,2) & \dots & P(1,N) \\ P(2,1) & P(2,2) & \dots & P(2,N) \\ \dots & \dots & \dots & \dots \\ P(K,1) & P(K,2) & \dots & P(K,N) \end{bmatrix} \tag{3}$$

While the joint probabilities of senones and Gaussians are generally not easy to compute, they can be estimated from a training corpus. The EM algorithm for learning HMM parameters estimates expected joint counts for senones and Gaussians, $n(k,s) = |T|w(k,s)$, where $|T|$ is the size of the training corpus. In fact, in most speech recognition systems, including the CMU Sphinx that we have used for our experiments, it is the $n(k,s)$ terms that are stored in the recognizer. Mixture weights are computed from them as

$$w_{k,s} = \frac{n(k,s)}{\sum_k n(k,s)} \tag{4}$$

The matrix P can be constructed by arranging the $n(k,s)$ terms into the appropriate $K \times N$ matrix and normalizing it such that all terms sum to 1.0.

3. LATENT VARIABLE DECOMPOSITION

Adaptation of the HMMs requires updating of all $K \times N$ entries in P. Typically the available adaptation are not sufficient to adapt all of the terms in P. Also, data available for some states may be much lesser than for others. We would like to adapt a small number of parameters using the adaptation data, in such a manner that all $K \times N$ terms are modified to better represent the test data. To do this, we decompose P into the following product form:

$$P = U\Sigma V^T \tag{5}$$

where U is an $K \times D$ matrix, Σ is an $D \times D$ diagonal matrix and V is a $N \times D$ matrix. Adaptation could be achieved by modifying either the D diagonal entries of Σ or the $K \times D$ entries of $U\Sigma$, both which are far fewer than the full $K \times N$ entries of P. D is the order of the decomposition. If it is lesser than K, the decomposition is lossy, otherwise it can be exact.

Decompositions of the form shown in Equation (5) can be obtained from singular value decomposition (SVD) of P. However SVD does not consider the fact that all entries of P are positive, and adaptation based on SVD decompositions can result in negative estimates for probabilities - an obvious anomaly. Instead, we use non-negative factorizations of the P matrix. This ensures that all elements of the U, Σ and V matrices are positive, thereby guaranteeing that their product is positive. Non-negative factorizations can be obtained using the NMF algorithm proposed by Tsung *et. al.* [3], or the PLSA algorithm proposed by Hoffman [4]. Of the two, the PLSA algorithm is more suited to our purpose since the decomposition inherently assumes that the P matrix is a probability matrix.

Under the PLSA model, the joint probability of Gaussians and senones can be expressed as:

$$P(k,s) = \sum_{d=1}^{D} P(\sigma = \sigma_d)P(k|\sigma)P(s|\sigma) \tag{6}$$

where σ is the latent variable that governs both the *a priori* probabilities of the Gaussians and the *a priori* probabilities of the senones. σ is permitted to take one of D values, $\sigma_1 \dots \sigma_D$. Given the entire set of $P(k,s)$ values, all $P(\sigma = \sigma_d)$, $P(k|\sigma)$ and $P(s|\sigma)$ terms are obtained using an expectation maximization algorithm [4]. Once these terms are obtained, they can be arranged in matrices in the form given in Equation (5), where $P(\sigma = \sigma_d)$ is the d^{th} diagonal element of Σ, $P(k|\sigma_d)$ is the $(k,d)^{th}$ element of U and $P(s|d)$ is the $(s,d)^{th}$ element of V.

Figure 1 illustrates the PLSA decomposition graphically. Fig. 1(a) shows the relationship between Gaussians and senones as given by P. The nodes on the left, labeled X, represent the various Gaussians, and those on the right, labeled Y, show the senones. Every edge represents the joint probability of a senone and a Gaussian. Note that this is not the conventional dependency graph illustration of Graphical models, since every probability entry has its own edge in the figure. Fig. 1(b) shows the PLSA decomposition of the distribution represented by Fig. 1(a). PLSA decomposition introduces a latent variable, which can take only D values. These are shown by the central "latent" layer of nodes in the graph, labeled Z. In Fig. 1(b), edges represent the conditional probability of senones or Gaussians, given the latent variable. The nodes for the latent variable themselves carry the *a priori* probability of the latent variable. The joint probability of any Gaussian

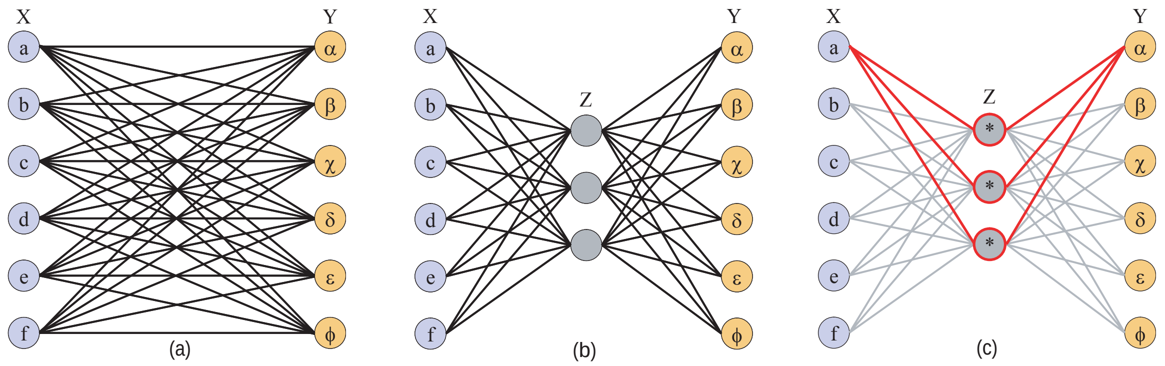

Figure 1: Graphical illustration of PLSA decomposition. (a) A graphical representation of a conventional 2-variable distribution. Each edge represents a joint probability of two specific values of the variables. (b) In a PLSA decomposition, the probability of each of the two variables is related only to the value of an intermediate latent variable. (c) The joint probability of two specific values of the variables is the total probability of all paths linking the two values.

and any senone is the sum of all the probabilities associated with all paths from the Gaussian to the senone. Fig. 1(c) illustrates this with an example of a single Gaussian-senone pair. The sum of the probabilities of all the darkened paths represents the joint probability of the Gaussian a and the senone α. The total number of edges in the graph represent the total number of free parameters in it. By appropriate selection of D, the total number of edges in Fig. 1(b) can be made significantly lesser than that in Fig. 1(a).

3.1 Maximum likelihood adaptation of mixture weights

The goal of adaptation is to modify every entry in the P matrix, based on some adaptation data, to best represent the expected conditions of the adaptation data. Direct adaptation would require updating all $K \times N$ terms in P. The amount of adaptation data available is often insufficient to update all $K \times N$ terms. On the other hand the U, Σ and V matrices obtained from PLSA decomposition of P have only $K \times D$, D and $N \times D$ non-zero terms respectively. Adaptation of all the entries in any one or more of these matrices results in adaptation of all $K \times N$ entries of P. The advantages of the decomposition become more apparent from Figure 1(c). In order to update all entries in P, it is sufficient to update any subset of probabilities such that their corresponding edges (or nodes, in the case of the latent variable) lie on at least one of the paths from every Gaussian node to every senone node. Thus, for example, in order to update $P(a, \alpha)$, it would be sufficient to update the probabilities of any of the edges on any of the paths that connect the Gaussian a and the senone α. Latent variable decomposition can thus be viewed as a framework that provides a facility for adapting entire joint distributions by adapting only a small number of probability terms.

The actual set of parameters can be selected based on the amount of adaptation data available (if sufficient data are available, it is best to adapt the entire P matrix). In our work we have explored the adaptation of U, Σ, V, and product terms such as U, Σ. The sizes of these matrices can be modified by modifying D. Larger values of D represent more accurate decompositions of P; however, the corresponding U, Σ and V matrices are also larger and require more data to adapt effectively. On the other hand, while smaller values of D result in coarser decompositions of P. adaptation of the component matrices requires lesser data. While the decomposition based adaptation can also be performed using MAP estimation, here we choose to use ML estimation.

ML adaptation of the matrices would ideally be incorporated into the Baum-Welch reestimation algorithm. However, the corresponding update equations are complicated and difficult to implement. Instead, we use a simpler approach. From the adaptation data we compute the expected counts $\hat{n}(k, s)$ for every Gaussian-senone pair using the Baum-Welch algorithm. The normalized expected counts are arranged in a $K \times N$ matrix $\hat{P}$. In order to adapt any component matrix, for instance the matrix U, we estimate the updated U as

$$\hat{U} = \arg\min_{\Omega}\{KL(P, \Omega\Sigma V^T)\} \tag{7}$$

where Σ and V are the matrices obtained by the decomposition of the original P, and $KL(P, Q)$ is the Kullback-Leibler distance between the distributions P and Q. The estimation of $\hat{U}$ using Equation (7) can be shown to be identical to the maximum likelihood estimation of $\hat{U}$ given the adaptation data, and can be performed using a reduced version of the EM algorithm that is used for PLSA decomposition. Once $\hat{U}$ is estimated, the adapted joint probabilities of Gaussians and senones is computed as

$$P_{adapted} = \hat{U}\Sigma V^T \tag{8}$$

Adapted mixture weights are obtained by normalizing the columns of $P_{adapted}$ to sum to 1.0.

Adaptation of other matrices, such as Σ, or V, or various product terms can be performed in a very similar manner.

3.2 Extension to multi-stream semi-continuous HMMs

The above exposition is based on the assumption that the state output distribution of any tied state is given by a single mixture of Gaussians. In several existing systems that use semi-continuous HMMs (e.g, Sphinx-2), state output distributions are defined on N (N>1) feature streams (4 in sphinx-2) and the likelihood values are the products of N independent Gaussian mixtures. In this scenario, a separate joint probability matrix P is constructed for each of the feature streams. Decomposition and adaptation of the various feature streams is done independently of each other.

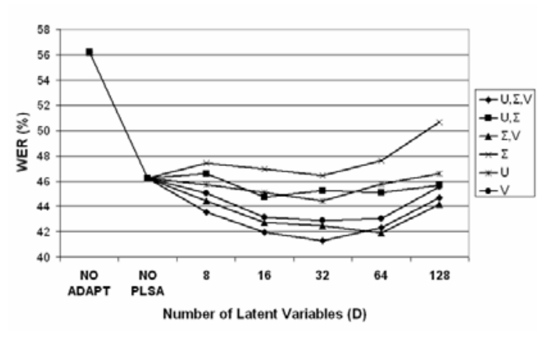

Figure 2. Variation of word error rate with the number of latent variables in the PLSA experiments.

4. EXPERIMENTS

The technique described in this paper was evaluated on the system developed at CMU for the project “Let's Go! Improved speech interfaces for the general public”. Using a telephone-based bus information system, this project aims to improve spoken dialogue systems by targeting two specific user populations: non-native speakers and elderly people. The proposed technique was used to adapt the acoustic models used in Let’s Go! to non-native speakers with a variety of accents.

The Let's Go! speech recognition system uses Sphinx-2, a large-vocabulary continuous speech recognition system based on semi-continuous HMMs. The acoustic models used for our experiments were semi-continuous HMMs with a codebook of 256 Gaussians, with 4000 senones. The base HMMs were trained on 102114 utterances (52 hrs.) of clean telephone speech from both native and, to a lesser extent, non-native users of the CMU Communicator system.

The test and adaptation data were collected between Feb. and Nov. 2003 using the Let’s go! system. The major part of the data was gathered through directed experiments where the subjects were given explicit tasks they had to complete using the system. Also, we used data from spontaneous calls to the system that were subsequently manually transcribed and labeled as native or non-native. In all, 3729 utterances (189 mins.) were transcribed. These data were partitioned into a test set of 449 utterances (20 mins.) from non-native speakers, and an adaptation set comprising 3280 utterances (169 mins.) from non-native speakers.

The baseline WER of the original acoustic models on the test set was 56.3%. In contrast, the same models yielded a 23.6% WER on a set of 452 utterances from calls by native speakers.

For our experiments, we first retrained our models on the adaptation data using the Baum-Welch algorithm initialized with the Communicator models. Then we applied PLSA adaptation for different values of D by adapting different sets of matrices. The resulting WERs on the test set are shown in Figure 2.

After Baum-Welch retraining, the models (labeled “NO PLSA” on Fig. 2) already performed significantly better than the original ones (NO ADAPT), with a WER of 46.3%. When applying PLSA adaptation, experiments showed that adapting V, which matches senones to latent variables, brings the most improvement in performance. Indeed, adapting Σ alone systematically resulted in a degradation in WER for any number of latent variables. Adapting U alone or in combination with Σ brought only a small improvement over the NO ADAPT models.

In terms of number of latent variables, for the best three matrix combinations (V, Σ V, UΣ V), WER improves with D up to 32 or 64, and degrades for higher values. These values may be specific to our adaptation set, which is both large (for an adaptation set) and heterogeneous (it contains multiple speakers with different accents). Further experiments are needed to study how these values vary with the size and homogeneity of the adaptation set.

Overall, the best result obtained by adapting all three matrices Σ , U and V, with 32 latent variables. The latter yielded a WER of 41.3%, a 26% relative improvement over the NO ADAPT baseline and 11% over the NO PLSA models.

5. DISCUSSION

The experiments show that the PLSA-based adaptation technique is effective at improving WER. The experiments reported in this paper utilize supervised adaptation of the models. In principle, unsupervised adaptation is also possible in the same manner. This will be part of future work.

The PLSA model trades off the resolution in the state output distributions with the robustness with which they can be estimated**:** reducing the number of latent variables reduces the resolution of the distributions, but improves the robustness with which distribution parameters can be estimated with small amounts of training data. A superior approach might be to interpolate between the original high-resolution models and the decomposed low-resolution models using a technique such as deleted interpolation. Future research will explore this possibility.

Experiments also indicate that the decomposition and reconstruction procedure itself introduces a degree of regularization that improves recognition of test data. Future research will also investigate this phenomenon.

ACKNOWLEDGEMENTS

This work was done under NSF Grant. No. IIS-0208835. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

REFERENCES

[1] Hwang, M. Y. and Huang, X. (1993). “Shared-distribution hidden Markov models for speech recognition”, IEEE Trans. Speech Audio Processing, Vol 1., pp. 414-420.

[2] Huo Q., Chan, C. and Lee, C.H. (1995), “Bayesian adaptive learning of the parameters of hidden Markov model for speech recognition”, IEEE Trans. Speech and Audio Proc., Vol. 3, pp 334-345.

[3] Lee, D.D. and Seung, H.S. (2001). “Algorithms for non-negative matrix factorization”, in Advances in Neural Information Processing Systems 13, pp. 556-562, MIT Press.

[4] Hoffman, T. (1999) “Probabilistic latent semantic analysis”, In proceedings of 15th Conference on Uncertainty in Artificial Intelligence, 289-296.