TR-IT-0124

Probabilistic Transfer Vector Prediction for Speaker Adaptation

ヘーズン ティモシィ
Timothy J. Hazen

1995.08

This report introduces a novel speaker adaptation algorithm, probabilistic transfer vector prediction (PVP), which is intended to be utilized as a supplement or alternative to transfer vector field smoothing (VFS) within a hidden Markov model (HMM) recognition system.

目次

1 Introduction

To perform fast speaker adaptation within a speech recognition system it is necessary to adapt the model parameters of the system based on a limited amount of adaptation data. This includes the model parameters of classes for which no adaptation data has been encountered. To do this, the adaptation of these parameters must be performed using only the observations available from other classes. Transfer vector field smoothing (VFS) is one method for performing this task which has proven successful [2, 4]. VFS has been shown to provide fast speaker adaptation of the mean vectors of a continuous mixture density hidden Markov model (CDHMM). However, the main assumption behind VFS adaptation is that parameter transfer vectors for states in which no adaptation data exists can be linearly interpolated from the transfer vectors of acoustically similar states. While this approach has been shown to be successful, the current research investigates a probabilistic approach in which the transfer vectors of states with no adaptation data are predicted using *a priori* knowledge about the correlations between the transfer vectors of different states.

This paper introduces probabilistic transfer vector prediction (PVP) as one method for incorporating a predictive model into the VFS framework. PVP utilizes *a priori* knowledge about the correlations which exist between states to be able to predict the transfer vector of one state from the estimates of transfer vectors of other states. In its use of Gaussian models to capture *a priori* knowledge about correlations, PVP is similar to the extended maximum *a posteriori* probability (EMAP) approach [1]. However, several key differences between EMAP and PVP exist and are discussed in this paper.

2 Theory

2.1 Problem Statement

The goal in a probabilistic approach to speaker adaptation is to adapt the models utilized by a speech recognition system to best fit the characteristics of a particular speaker given a set of adaptation data. To begin, let λ represent a vector of model parameters that are used by the recognizer and are to be adjusted using adaptation data. If we assume that λ contains parameters which model C different classes then we can subdivide the vector λ into subvectors such that

$$\lambda = [\lambda_1^T , \lambda_2^T , \ldots , \lambda_C^T]^T \tag{1}$$

where each λ_j represents the vector of parameters used to model the j^{th} class. Next, let $\mathcal{X}$ be the set of adaptation data. In particular, let $\mathcal{X}$ be represented as

$$\mathcal{X} = \{X_1 , X_2 , \ldots , X_C\} \tag{2}$$

where each X_j is a set of example observations from the j^{th} class as spoken by the current speaker. In total there are C different classes. Furthermore, we will represent the sets of observations from each class as

$$X_j = \{\mathbf{x}_{j,1} , \mathbf{x}_{j,2} , \ldots , \mathbf{x}_{j,n_j}\} \tag{3}$$

where each $\mathbf{x}_{j,k}$ is a specific example vector of class j, and n_j is the number of adaptation examples for class j. Using the above definition the adaptation problem can be approached using a maximum *a posteriori* probability (MAP) algorithm which is based on the equation

$$\hat{\lambda} = \arg\max_{\forall\lambda} \mathrm{p}(\lambda \mid \mathcal{X}). \tag{4}$$

In this equation $\hat{\lambda}$ contains the adapted values of the model's parameters. Using Bayes rule the equation can be rewritten as:

$$\hat{\lambda} = \arg\max_{\forall\lambda} \frac{\mathrm{p}(\mathcal{X} \mid \lambda)\mathrm{p}(\lambda)}{\mathrm{p}(\mathcal{X})}. \tag{5}$$

By noting that the denominator in the equation is independent of λ the equation can be equivalently expressed as:

$$\hat{\lambda} = \arg\max_{\forall\lambda} \mathrm{p}(\mathcal{X} \mid \lambda)\mathrm{p}(\lambda). \tag{6}$$

Various different MAP approaches, including the one presented in this paper, utilize the equation above as the basis for their adaptation algorithm.

2.2 Standard MAP Approach

The most basic approach to speaker adaptation is known as the standard MAP approach. In this approach, two primary assumptions are made. First, each acoustic observation $\mathbf{x}_{j,k}$ is considered to be dependent only on the parameters of the model corresponding to its own class. Thus, each observation is considered independent of all other observations and independent of the parameters of all classes in the model except the class to which it belongs. This assumption allows Equation (6) to be rewritten as:

$$\hat{\lambda} = \arg\max_{\forall\lambda} \mathrm{p}(\lambda) \prod_{j=1}^{C} \prod_{k=1}^{n_j} \mathrm{p}(\mathbf{x}_{j,k} \mid \lambda_j). \tag{7}$$

The standard MAP approach also assumes that the parameters of each class are independent of each other. With this assumption Equation (7) can be reduced to

$$\hat{\lambda} = \arg\max_{\forall\lambda} \prod_{j=1}^{C} \mathrm{p}(\lambda_j) \prod_{k=1}^{n_j} \mathrm{p}(\mathbf{x}_{j,k} \mid \lambda_j). \tag{8}$$

Because of the assumption of independence between the classes, the parameters for each class' model can be adapted separately using the equation

$$\hat{\lambda}_j = \arg\max_{\forall \lambda_j} \mathrm{p}(\lambda_j) \prod_{k=1}^{n_j} \mathrm{p}(\mathbf{x}_{j,k} \mid \lambda_j). \tag{9}$$

If we assume that for a given class, the likelihood of an observation is represented by a Gaussian density function then we can write

$$\mathrm{p}(\mathbf{x}_j \mid \lambda_j) \equiv \mathcal{N}(\mu_j, \mathbf{S}_j) \tag{10}$$

where μ_j and $\mathbf{S}_j$ represent the mean vector and covariance matrix for the Gaussian density function.

Suppose we are interested in adapting only the mean parameters of a Gaussian model. In such a case we need to know the *a priori* density function of the mean vector for any random speaker as well. Let this be represented as

$$\mathrm{p}(\mu_j) \equiv \mathcal{N}(\mu_{0j}, \mathbf{S}_{0j}) \tag{11}$$

where μ_{0j} and $\mathbf{S}_{0j}$ are the *a priori* mean vector and covariance matrix for a random speaker's parameter space. Using the above assumptions it can be shown that the adapted value $\hat{\lambda}_j$ that maximizes the *a posteriori* probability density function can be represented as:

$$\hat{\lambda}_j = \mathbf{S}_{0j} \left(\mathbf{S}_{0j} + \frac{\mathbf{S}_j}{n_j} \right)^{-1} \bar{\mathbf{x}}_j + \frac{\mathbf{S}_j}{n_j} \left(\mathbf{S}_{0j} + \frac{\mathbf{S}_j}{n_j} \right)^{-1} \mu_{0j} \tag{12}$$

where $\bar{\mathbf{x}}_j$ is the maximum likelihood (ML) estimate of the mean vector as estimated from the adaptation data using the equation

$$\bar{\mathbf{x}}_j = \frac{1}{n_j} \sum_{k=1}^{n_j} \mathbf{x}_{j,k}. \tag{13}$$

By examining Equation (12) it can be seen that the MAP estimate of a mean vector is simply an interpolation between the *a priori* mean and the ML estimate of the mean from the adaptation data. As the number of adaptation observations increases and the ML estimate becomes more reliable, the adapted mean vector shifts away from the *a priori* mean towards the ML estimate. Because of this basic result the MAP estimation equation is often approximated with the simpler equation

$$\hat{\lambda}_j = \frac{n_j \bar{\mathbf{x}}_j + \tau \mu_{0j}}{n_j + \tau} \tag{14}$$

where τ is a scalar (or possibly matrix) which is experimentally found to optimize the speaker adaptation performance on held out data.

Standard MAP estimation has optimal asymptotic performance, i.e. with standard MAP estimation the model parameters will converge to the ML estimate of the speaker's parameter space as the amount of adaptation data increases towards infinity. However, because the algorithm does not take advantage of correlations which might exist between the parameters of different classes, the algorithm can only provide good estimates of the parameters of classes for which it has observed an adequate amount of examples. Because of this limitation, the algorithm is slow in adapting to a new speaker's parameter space and significant improvements in the model's parameters may require many adaptation sentences.

2.3 Extended MAP Approach

To take advantage of the correlations that may exist between parameters from different classes across the space of all speakers, the extended MAP (EMAP) estimation algorithm was proposed [1]. To describe EMAP begin with Equation (7):

$$\hat{\lambda} = \arg\max_{\forall \lambda} \mathrm{p}(\lambda) \prod_{j=1}^{C} \prod_{k=1}^{n_j} \mathrm{p}(\mathbf{x}_{j,k} \mid \lambda_j). \tag{15}$$

In this equation, standard MAP estimation assumes that the parameters for each separate class in the $p(\lambda)$ are independent. EMAP does not use this assumption. Instead EMAP maintains the original form of Equation (7) in order to capture the correlations between the classes within the $p(\lambda)$ term.

If we again utilize Gaussian density functions to model the different $p(\mathbf{x}_{j,k} \mid \lambda_j)$ terms, EMAP is the same as MAP in its use of the expression

$$p(\mathbf{x}_j \mid \lambda_j) \equiv \mathcal{N}(\mu_j, \mathbf{S}_j). \tag{16}$$

However, in EMAP, instead of using a separate *a priori* model for the mean vector's λ_j for each state, the generalized vector λ is modeled with one single Gaussian density function incorporating the parameters from all classes. Thus, we utilize the expression

$$p(\mu) \equiv \mathcal{N}(\mu_0, \mathbf{S}_0) \tag{17}$$

where

$$\mu = [\,\mu_1^T\,,\,\mu_2^T\,,\,\cdots\,\mu_C^T\,]^T \tag{18}$$

and where

$$\mathbf{S}_0 = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} & \cdots & \mathbf{S}_{1C} \\ \mathbf{S}_{21} & \mathbf{S}_{22} & \cdots & \mathbf{S}_{2C} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_{C1} & \mathbf{S}_{C2} & \cdots & \mathbf{S}_{CC} \end{bmatrix}. \tag{19}$$

From this equation we can deduce that EMAP is identical to MAP if, within $\mathbf{S}_0$, we set $\mathbf{S}_{ij} = 0$ when $i \neq j$. Using the above definitions, the adaptation of a model's Gaussian mean vector is achieved with the expression

$$\hat{\lambda} = \mathbf{S}_0\,(\mathbf{N}\mathbf{S}_0 + \mathbf{S})^{-1}\,\mathbf{N}\bar{\mathbf{x}} \;+\; \mathbf{S}\,(\mathbf{N}\mathbf{S}_0 + \mathbf{S})^{-1}\,\mu_0 \tag{20}$$

where

$$\bar{\mathbf{x}} = [\,\bar{\mathbf{x}}_1^T\,,\,\bar{\mathbf{x}}_2^T\,,\,\cdots\,,\,\bar{\mathbf{x}}_C^T\,]^T \tag{21}$$

and

$$\mathbf{N} = \begin{bmatrix} n_1\mathbf{I} & 0 & \cdots & 0 \\ 0 & n_2\mathbf{I} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & n_C\mathbf{I} \end{bmatrix} \tag{22}$$

and

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{S}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{S}_C \end{bmatrix}. \tag{23}$$

Theoretically, EMAP has nice properties in that it accounts for the correlations between the parameters of different classes and the adapted parameters converge asymptotically to their ML estimated as the amount of adaptation data is increased. However, in practice it is very difficult to gain advantage from using the EMAP algorithm because of the size of the correlation matrix that must be trained. If there are C classes and each class has a D dimensional parameter vector then the matrix $\mathbf{S}_0$ is a $CD \times CD$ size matrix. In order for this matrix to even be invertible a total of $CD + 1$ training vectors must be used in training. In other words, a corpus containing CD different speakers must be available to prevent $\mathbf{S}_0$ from being singular. Additionally, an estimate for the mean of each class for all speakers is also necessary in training. Thus, spontaneous speech databases in which each speaker may not have uttered examples of all classes will be difficult to use for training the EMAP covariance matrix.

In practice, simplifying assumptions can be made which reduce the number of covariance elements in $\mathbf{S}_0$ which must be trained. For example, only the diagonal elements of each submatrix $\mathbf{S}_{ij}$ can be trained with all other elements being set to zero [3, 5]. This simplification reduces the amount of training data necessary to sufficiently train $\mathbf{S}_0$ but it does not remove the requirement that an estimate of the mean vector for all classes must be made for all speakers.

2.4 MAP-VFS

Transfer vector field smoothing using maximum *a posteriori* probability estimation (MAP-VFS) is based on the assumption that acoustically similar classes can be adapted in similar fashions [2, 4]. In particular let us define a *transfer vector* as the difference between the *a priori* parameter vector and an estimate of the same parameter vector for one particular speaker. Mathematically, we can express this as

$$\mathbf{v}_i = \lambda_{i0} - \lambda_i \tag{24}$$

where λ_i is the speaker specific estimate of the parameter vector and $\mathbf{v}_i$ is the transfer vector.

The MAP-VFS algorithm is a three step process with the following steps:

1. Estimate the transfer vectors for classes present in the adaptation data using standard MAP estimation.
2. Create transfer vectors for classes which have not been encountered in the adaptation data through interpolation of acoustically similar classes for which transfer vector estimates exist.
3. Smooth the transfer vectors to help compensate for estimates which may be poorly trained due to insufficient amounts of adaptation data.

MAP-VFS has been shown to be effective for fast speaker adaptation [4]. It's success comes from the fact that the interpolation and smoothing portions of the algorithm are able to adapt the parameters of a particular class' model even without having seen examples of that class. By assuming that the parameters of acoustically similar classes will be adapted in similar fashions many classes can be adapted based on the observations of only a few classes. This algorithm also has the advantage that it has no models which require prior training and only a few parameters that need to be optimized. However, the assumptions in VFS may also leave room for improvement in the algorithm. It may be possible to improve the VFS adaptation scheme by introducing probabilistic prediction as an alternative to the interpolation scheme.

2.5 Probabilistic Transfer Vector Prediction

Probabilistic transfer vector prediction (PVP) is intended to be a compromise between EMAP adaptation and MAP-VFS adaptation. The general idea behind PVP is to utilize the correlations inherent between classes to guide the adaptation of classes for which little or no adaptation data has been seen. MAP-VFS uses no *a priori* information about these correlations, while EMAP assumes a large global correlation structure covering all classes which in practice is difficult to train. PVP attempts to utilize the same correlations between classes without requiring a global correlation structure.

To understand the basis of PVP, begin with the generic MAP expression

$$\hat{\lambda} = \arg\max_{\forall\lambda} \mathrm{p}(\lambda)\frac{\mathrm{p}(\mathcal{X} \mid \lambda)}{\mathrm{p}(\mathcal{X})}. \tag{25}$$

Next, expand the expression out to the individual observations by assuming all observations are independent of each other to yield the expression

$$\hat{\lambda} = \arg\max_{\forall\lambda} \mathrm{p}(\lambda)\prod_{j=1}^{C}\prod_{k=1}^{n_j}\frac{\mathrm{p}(\mathbf{x}_{j,k} \mid \lambda)}{\mathrm{p}(\mathbf{x}_{j,k})}. \tag{26}$$

Up to this point EMAP and PVP have utilized the same assumptions. However, from this equation EMAP makes the assumption that observations of each class are independent of all model parameters except the parameters of their own class, while at the same time not making any simplifying assumptions about the correlations between the models parameters themselves. Theoretically, these assumptions are generally considered sound. PVP, on the other hand, takes a slightly different approach.

First, the expression can be rewritten using Bayes rule to yield the equation:

$$\hat{\lambda} = \arg\max_{\forall\lambda} \mathrm{p}(\lambda)\prod_{j=1}^{C}\prod_{k=1}^{n_j}\frac{\mathrm{p}(\lambda \mid \mathbf{x}_{j,k})}{\mathrm{p}(\lambda)}. \tag{27}$$

Next, we assume that the parameters of each class are independent of the parameters of all other classes. Thus, we can rewrite the maximization process by considering each class independently as follows:

$$\hat{\lambda}_i = \arg\max_{\forall \lambda_i} \mathrm{p}(\lambda_i) \prod_{j=1}^{C} \prod_{k=1}^{n_j} \frac{\mathrm{p}(\lambda_i \mid \mathbf{x}_{j,k})}{\mathrm{p}(\lambda_i)}. \tag{28}$$

The final expression in Equation (28) shows an adaptation scheme in which the models begin with the *a priori* models and are readapted with each new observation. Each new observation produces a model $\mathrm{p}(\lambda_i \mid \mathbf{x}_{j,k})$ which predicts a new value for each parameter λ_i based the correlation between observations of class j and the model parameters for class i. The ratio between the density function for the new prediction of λ_i and the *a priori* prediction of λ_i is used to update the current prediction of λ_i.

Using this approach, the classes in which little or no adaptation data is available can be adapted using observations of other classes and *a priori* correlation information. On the other hand, the assumptions that are made also prevent the parameters of a class from ever asymptotically converging to the ML estimate when sufficient data from a particular class is available. This is because the assumption that the observations of a class are dependent only on the parameters of its class was not made, as it was in both the MAP and EMAP formulations. However, if this method of adaptation is combined with the standard MAP adaptation algorithm using an interpolation scheme which shifts the weight of a class' estimate from the PVP estimate to the MAP estimate as more examples of a particular class are seen, then the proper asymptotic properties can also be achieved.

3 Gaussian Modeling with PVP

3.1 Models Utilized in PVP

To utilize PVP, models must exist for the expressions $p(\lambda_i)$ and $p(\lambda_i, \mathbf{x}_j)$. The term $p(\lambda_i)$ is the *a priori* probability density function of a random speaker possessing the model parameters λ_i for class i. The term $p(\lambda_i, \mathbf{x}_j)$ is the joint probability density function of a random speaker possessing the model parameters λ_i for class i and producing the observation vector $\mathbf{x}_j$ as a random example of class j. The joint probability expression is needed in order to calculate $p(\lambda_i \mid \mathbf{x}_j)$ when a given observation $\mathbf{x}_j$ is provided. Both of these models must by created from a training corpus. While any parametric density function can theoretically be used to model these expressions, the remainder of this section will assume that the models are standard multivariate Gaussian density functions.

3.2 Training the *a priori* Model

The model $p(\lambda_i)$ is designed to capture the *a priori* likelihood of a random speaker possessing the model parameters λ_i for class i. This model can be created by estimating the values of λ_i for every speaker in a training corpus and using this collection of λ_i training vectors to estimate the parameters for $p(\lambda_i)$. This training will result in a Gaussian density function which we will define as follows:

$$p(\lambda_i) \equiv \mathcal{N}(\mu_{i0}, \mathbf{S}_{i0}). \tag{29}$$

3.3 Training the Joint Probability Model

The model $p(\lambda_i, \mathbf{x}_j)$ is designed to capture the joint likelihood of observing a random observation $\mathbf{x}_j$ from class j as spoken by a random speaker possessing model parameters λ_i for class i. To train the model, we must create a set of joint probability vectors for each speaker. Each vector must contain the mean vector of class i (i.e., λ_i) and an observation of class j (i.e., $\mathbf{x}_j$). One joint vector is created for each example of $\mathbf{x}_j$ within the training set for each training speaker. The accumulation of all of these vectors can then be used to train the model $p(\lambda_i, \mathbf{x}_j)$. Figure 1 shows an illustration of the training for this model. In this illustration there are vectors from 4 speakers with each speaker contributing 7 data points. For all data points for a particular speaker the mean vector λ_i remains the same but the observations of $\mathbf{x}_j$ change. By building a Gaussian model of the conglomeration of all vectors from all speakers we hope to capture the general correlations between λ_i and $\mathbf{x}_j$ across all speakers in the training set.

There is one major training problem that must be addressed: what should be done about the varying sizes of the sets of $\mathbf{x}_j$ vectors for each speaker. Ideally we would like to have identical numbers of $\mathbf{x}_j$ examples for each speaker, thus allowing each speaker to contribute equally in the training of the model. The two most obvious methods of training are: (1) ignore the disparities that exist in the amounts of data from each speaker and train the model with all training vectors weighted equally, or (2) weight all vectors from the same speaker by the inverse of the total number of vectors from that speaker. Option (1) could cause the model to be dominated by the speakers with the most data while option (2) allows the example data points from speakers with fewer examples to be weighted more heavily than the data points from speakers with more data. Estimation problems will likely arise from either solution unless a very large amount of data from all speakers is available. For the experiments discussed later in this paper, option (2) was used for training.

In the end the trained model can be represented as:

$$p(\lambda_i, \mathbf{x}_j) \equiv \mathcal{N}(\mu_{ij}, \mathbf{S}_{ij}). \tag{30}$$

In discussing this model we will use the definitions

$$\mu_{ij} = \begin{bmatrix} \mu_{\lambda_i} \\ \mu_{x_j} \end{bmatrix} \tag{31}$$

and

$$\mathbf{S}_{ij}^{-1} = \begin{bmatrix} (\mathbf{S}_{ij}^{-1})_{\lambda\lambda} & (\mathbf{S}_{ij}^{-1})_{\lambda x} \\ (\mathbf{S}_{ij}^{-1})_{x\lambda} & (\mathbf{S}_{ij}^{-1})_{xx} \end{bmatrix}. \tag{32}$$

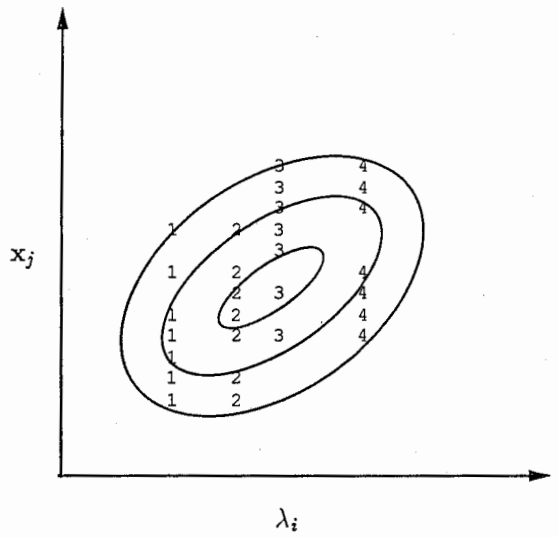

図 1: Illustration of example set of vectors used to train jointly Gaussian expression $p(\lambda_i, \mathbf{x}_j)$

Within these definitions it can be noted that $\mu_{i0} = \mu_{\lambda_i}$.

3.4 Adaptation Calculations

In practice we are interested in using the conditional density function $p(\lambda_i \mid \mathbf{x}_j)$, and not the joint density function $p(\lambda_i, \mathbf{x}_j)$. However, given the conditional value $\mathbf{x}_j$ the conditional density function can be easily found from the joint density function. The conditional density function is also Gaussian and can be described as

$$p(\lambda_i \mid \mathbf{x}_j) \equiv \mathcal{N}(\mu_{\lambda_i|x_j}, \mathbf{S}_{\lambda_i|x_j}). \tag{33}$$

where

$$\mathbf{S}^{-1}_{\lambda_i|x_j} = (\mathbf{S}^{-1}_{ij})_{\lambda\lambda} \tag{34}$$

and

$$\mu_{\lambda_i|x_j} = \mu_{\lambda_i} + (\mathbf{S}^{-1}_{ij})^{-1}_{\lambda\lambda}(\mathbf{S}^{-1}_{ij})_{\lambda x}(\mu_{x_j} - \mathbf{x}_j). \tag{35}$$

Given the expressions for $p(\lambda_i)$ and $p(\lambda_i \mid \mathbf{x}_j)$, we can generate an expression for $p(\lambda_i \mid \mathcal{X})$ using the equation

$$p(\lambda_i \mid \mathcal{X}) = p(\lambda_i)^{(1-N)} \prod_{j=1}^{C} \prod_{k=1}^{n_j} p(\lambda_i \mid \mathbf{x}_{j,k}) \tag{36}$$

where

$$N = \sum_{j=1}^{C} n_j. \tag{37}$$

From this equation we find that $p(\lambda_i \mid \mathcal{X})$ is also a Gaussian which can be defined as

$$p(\lambda_i \mid \mathcal{X}) \equiv \mathcal{N}(\mu_{i|\mathcal{X}}, \mathbf{S}_{i|\mathcal{X}}) \tag{38}$$

where

$$\mathbf{S}^{-1}_{i|\mathcal{X}} = (1-N)\mathbf{S}^{-1}_{i0} + \sum_{j=1}^{C} n_j(\mathbf{S}^{-1}_{ij})_{\lambda\lambda} \tag{39}$$

and

$$\mu_{i|\mathcal{X}} = \mu_{i0} + \mathbf{S}_{i|\mathcal{X}} \left[\sum_{j=1}^{C} n_j(\mathbf{S}^{-1}_{ij})_{\lambda x}(\mu_{j0} - \bar{\mathbf{x}}_j) \right]. \tag{40}$$

Thus, the final PVP adaptation expression is

$$\hat{\lambda}_i = \arg\max_{\forall \lambda_i} \mu_{i|\mathcal{X}}. \tag{41}$$

We can also express the adapted mean in terms of transfer vectors where the ML transfer vector $\bar{\mathbf{v}}_j$ is defined as

$$\bar{\mathbf{v}}_j = \bar{\mathbf{x}}_j - \mu_{j0}. \tag{42}$$

Thus, in terms of transfer vectors, Equation (40) can be expressed as

$$\mu_{i|\mathcal{X}} = \mu_{i0} + \hat{\mathbf{v}}_i \tag{43}$$

where $\hat{\mathbf{v}}_i$ is simply a predicted transfer vector which is expressed as

$$\hat{\mathbf{v}}_i = \mathbf{S}_{i|\mathcal{X}} \left[\sum_{j=1}^{C} n_j (\mathbf{S}_{ij}^{-1})_{\lambda x} (-\bar{\mathbf{v}}_j) \right]. \tag{44}$$

In examining the expression for $\mu_{i|\mathcal{X}}$ we see that with no example observations, $\mu_{i|\mathcal{X}}$ simply equals the *a priori* mean for the class, μ_{i0}. As adaptation data is introduced, then $\mu_{i|\mathcal{X}}$ is shifted away from the *a priori* mean by adding a predicted transfer vector $\hat{\mathbf{v}}_i$. The vector $\hat{\mathbf{v}}_i$ is calculated from weighted predictions of the transfer vector provided by the ML estimated transfer vectors of all classes that have already been seen. The weighting factors in this case are n_j and $(\mathbf{S}_{ij}^{-1})_{\lambda x}$. The $(\mathbf{S}_{ij}^{-1})_{\lambda x}$ term allows transfer vectors which are most highly correlated with the current class i to be weighted more heavily than those from classes less correlated with class i. The n_j term allows transfer vectors which have been estimated from larger amounts of data to be be more heavily weighted than than those with small numbers of examples for basing an estimate, i.e. the transfer vectors that are more accurately estimated are weighted more than the ones that are less accurately estimated.

3.5 Comparison of PVP Algorithm to VFS Algorithm

In some ways PVP is very similar to VFS. Both attempt to create new transfer vectors for unseen classes based on the transfer vectors which have been estimated from the available adaptation data. Both, do not solely rely on the ML estimated transfer vector for classes in which data has been seen; rather, both perform a form of "smoothing" using the transfer vectors of other classes. Despite the similarities the three major differences that exist are:

1. VFS does not *predict* the transfer vectors of unseen classes but rather just *interpolates* them from acoustically similar classes.

2. VFS uses an acoustic distance between two classes rather than a correlation matrix to determine the influence or weight that one class' estimated transfer vector has in predicting another class' transfer vector.

3. Standard VFS does not consider the reliability of a transfer vectors' estimate when using it during vector smoothing. To compensate for this drawback, MAP-VFS was introduced. MAP-VFS uses the MAP estimates of the transfer vectors rather than the ML estimates during its interpolation and smoothing process.

3.6 Comparison of PVP Algorithm to EMAP Algorithm

In several ways PVP is very similar to EMAP. For one, both require that $(CD)^2$ correlation values be estimated. These correlation values also correspond to roughly the same measurements and hence both are attempting to capture the same information. Thus, both PVP and EMAP will suffer from a lack of sufficient training data to accurately estimate such a large collection of parameters. To compensate for this both require some form of parameter reduction to be effective. The major differences between PVP and EMAP are as follows:

1. In EMAP the trained correlation values all occupy one giant $CD \times CD$ correlation matrix while in PVP these values are distributed amongst C^2 separate DxD matrices. Thus EMAP assumes one giant *global* model while PVP contains many it local models.

2. The global nature of the EMAP correlation matrix requires than an estimate of the models parameters for all classes must be made for each speaker in order to be able to utilize that speaker's data during training. PVP does not have this requirement.

3. The EMAP parameter estimate will converge to the ML parameter estimate as the amount of adaptation data is increased. The PVP estimate needs an interpolation scheme in order to converge to the ML estimate.

4 Implementation Issues

4.1 Overview

In implementing the PVP algorithm there are several engineering issues that must be addressed. The issues stem from two primary problems. The first problem is that we currently do not have a sufficient amount of training data to estimate the entire set of $(CD)^2$ PVP correlation parameters accurately. Thus, ways must be found to reduce the number of correlation parameters that are trained and/or to compensate for the inaccurate predictions that may result from poorly estimated correlation matrices.

The second problem stems from the assumption that all frames are independent of each other. This assumption ignores the correlations which do exist between adjacent frames or frames within the same phoneme. In PVP, because each frame is used as a new observation, the system adapts too rapidly to the adaptation data because of this ignorance of the frame to frame correlations. Thus, ways must be found to adjust the adaptation rate to provide more reliable predictions.

4.2 Parameter Reduction

If the correlations between *all* C classes and *all* D measurements are utilized, parameter estimation problems will cause large difficulties. An HMM system with 50 states and 34 measurements would thus require $(50 \times 34)^2$ or 2.89 million parameters to be trained in the set of correlation matrices alone. To reduce the number of parameters to a more reasonable level two steps are taken.

First, consider the covariance matrix

$$\mathbf{S}_{ij} = \begin{bmatrix} (\mathbf{S}_{ij})_{\lambda\lambda} & (\mathbf{S}_{ij})_{\lambda x} \\ (\mathbf{S}_{ij})_{x\lambda} & (\mathbf{S}_{ij})_{xx} \end{bmatrix}. \tag{45}$$

Within each submatrix the strongest correlations exist along the diagonal elements. Thus by considering only the diagonal elements of each submatrix (and zeroing out all other covariance elements) we can reduce the number of elements that must be trained from $(CD)^2$ to $2DC^2$. For the 50 state HMM with 34 measurements this is a reduction from 2.89 million parameters to 170 thousand parameters.

Second, the correlations between all states need not be considered. Only the correlations between the most similar or most correlated states can be considered. One means of doing this is to consider only the correlations of the K-nearest neighbors to a class (including itself). Thus, with $K = 2$, only the correlations between a class and itself and its one nearest neighbor are considered. In this case the number of trained correlation parameters is reduced to $2DKC$, or 6,800 in the given example.

4.3 Adaptation Rate Control

In practice PVP suffers from overly rapid adaption. This is a result of the assumption that all frames are considered independent. PVP utilizes every new frame to readapt the model parameters. Because successive frames from the same observation of one class are highly correlated, each new frame from this phone may not be contributing any new information that is useful for adaptation. There are several ways to compensate for this problem. One potential method is to reduce the weight that each frame contributes to the adaptation.

In examining the equation

$$\mu_{i|\mathcal{X}} = \mu_{i0} + \mathbf{S}_{i|\mathcal{X}} \left[\sum_{j=1}^{C} n_j (\mathbf{S}_{ij}^{-1})_{\lambda x} (-\bar{\mathbf{v}}_j) \right] \tag{46}$$

it is seen that each ML estimated transfer vector is weighted by the number of frames used to estimate the transfer vector. By reducing this weight the adaptation can be slowed. The simplest way to do this is to multiply each count value n_j by a rate factor r where $0 < r \leq 1$. Thus each count value would be rerepresented as

$$\hat{n}_j = r\, n_j. \tag{47}$$

4.4 Model Interpolation

Despite the reduction in parameters, the PVP prediction may not be entirely reliable because of the potential of the correlation parameters being poorly estimated. In this case, it is desirable to avoid placing full faith in the PVP predicted transfer vectors, but rather interpolate the PVP predicted mean vectors with the *a priori* and/or ML estimated mean vectors. Thus, a weighted combination of the different estimates of the means can be defined as

$$\hat{\lambda} = w_0\lambda_0 + w_{pvp}\lambda_{pvp} + w_{ml}\lambda_{ml} \tag{48}$$

where

$$w_0 + w_{pvp} + w_{ml} = 1. \tag{49}$$

Thus, if $w_{pvp} = 0$ then the proper selection of w_0 and w_{ml} will simply yield the MAP estimate. Regardless, it is desirable to shift the estimate $\hat{\lambda}$ towards the ML estimate as more and more adaptation for each specific class becomes available.

5 Experimental Results

5.1 Experimental Platform

A series of experiments were performed using a simple HMM system to examine the capabilities of PVP. This preliminary set of experiments was conducted using a 50 state HMM phonetic recognition system where each state's output density function was modeled with a single diagonal Gaussian. The recognition system and the PVP *a priori* models were both trained using Japanese read speech collected from 291 different speakers. The adaptation of the HMM was performed by adapting the mean values of the Gaussian density functions of each state. The variances of the Gaussian density functions were not adapted.

Testing of the system was performed using data from seven adaptation speakers. Adaptation sets containing random selections of 1, 3, 5 and 7 adaptation phrases were created for each speaker. A separate test set containing 280 phrases per speaker was used for testing. Performance improvements were measured by the drop in phonetic recognition error rate.

5.2 Standard PVP

The first experiment was to examine standard PVP adaptation without utilizing any adaptation rate adjustment (i.e. $r = 1$) and without smoothing the PVP estimate with the *a priori* speaker independent model or the the ML model (i.e. $w_0 = 0$, $w_{pvp} = 1$ and $w_{ml} = 0$). The PVP performance was examined using 1, 3, 5, and 7 adaptation phrases per speaker as the number of K-nearest neighbor correlation matrices was varied. As can be seen in Figure 2, for each K recognition performance improves as the number of adaptation phrases is increased. Unfortunately, for all K the performance drops below the speaker independent (SI) performance after 1 adaptation utterance is observed. For $K = 1$, $K = 2$ and $K = 3$ the performance eventually surpasses the SI performance with more adaptation data. However, as K is increased beyond 2 the performance degrades significantly.

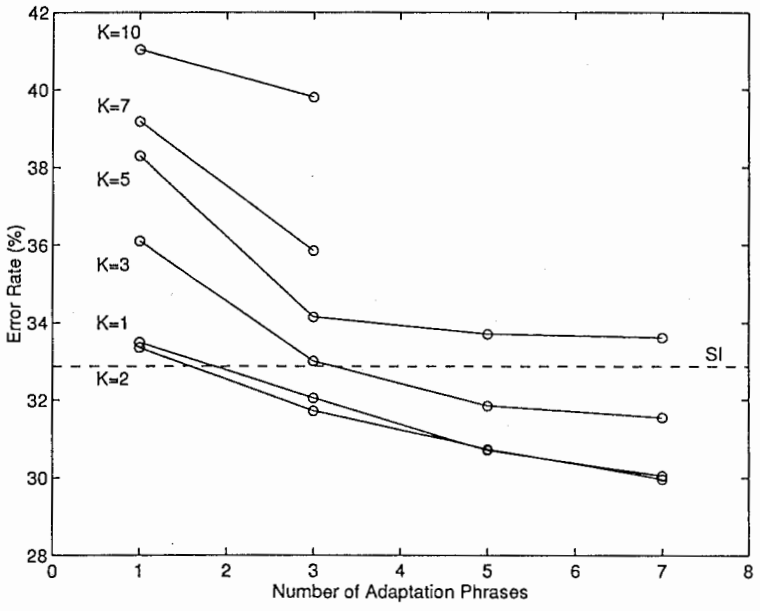

図 2: Phonetic recognition performance of standard PVP using 1, 3 ,5 and 7 adaptation phrases over varying values of K.

5.3 PVP with Adaptation Rate Control

It is believed that the poor performance PVP demonstrated in Figure 2 can be partially corrected by slowing down the adaptation rate of the system to prevent the system from adapting too quickly to poorly

estimated predictions. Figure 3 shows the performance of PVP using $K = 2$ as the adaptation rate r and the number of adaptation phrases is varied. As can be seen in the figure, the error rate can be reduced, especially when fewer adaptation phrases are utilized, by reducing the adaptation rate. The performance is also optimized by pushing the adaptation rate back up towards $r = 1$ as the number of adaptation phrases is increased.

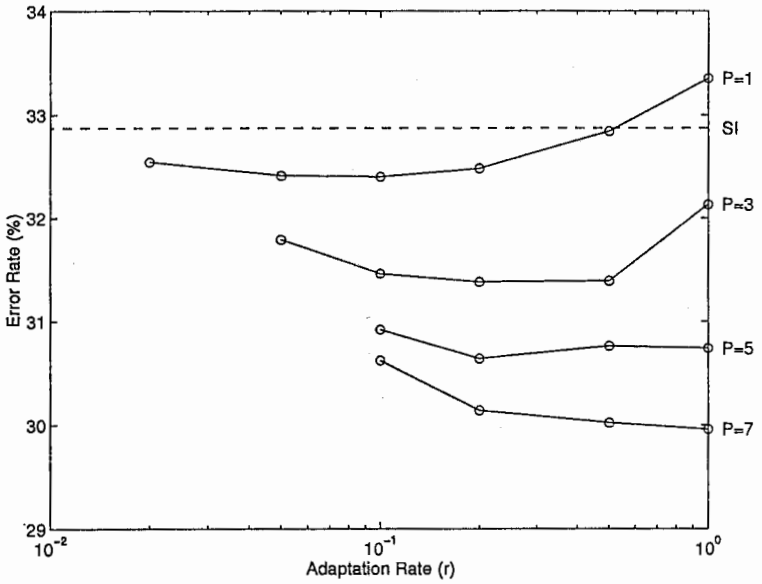

図 3: Phonetic recognition performance of PVP using $K = 2$ as the adaptation rate r and the number of adaptation phrases P is varied.

Figure 4 shows the performance of PVP when the adaptation rate is optimized as the value of K and the number of adaptation phrases is varied. As can be seen, optimizing the adaptation rate greatly improves the performance of the system over the standard PVP performance shown in Figure 2. However, PVP with $K = 2$ still performs better than PVP with $K > 2$ and only performs marginally better than PVP with $K = 1$.

5.4 PVP with Model Smoothing

As an alternative to utilizing an adaptation rate factor. PVP was tested using model smoothing. Smoothing was performed only with the *a priori* SI model (and not the ML model) using the following expression

$$\hat{\lambda} = (1 - w)\lambda_0 + w\lambda_{pvp} \tag{50}$$

where w is the weight for the PVP estimated model and the remainder of the weight is placed on the *a priori* SI model. Figure 5 shows the performance of PVP for $K = 2$ as the smoothing weight w and the number of adaptation phrases P are varied. As can be seen in the figure, the PVP performance can be improved more using optimized smoothing weights than with using optimized adaptation rates as presented in Figure 3. Experiments also showed that combining the adaptation rate and smoothing weights together within the system performs worse than only using that smoothing weight factors.

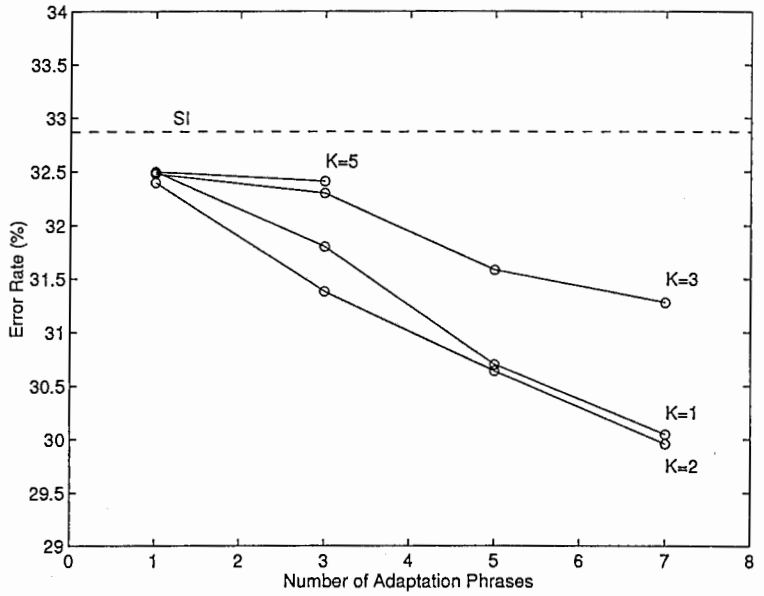

図 4: Phonetic recognition performance of PVP when the adaptation rate is optimized as the value of K and the number of adaptation phrases are varied.

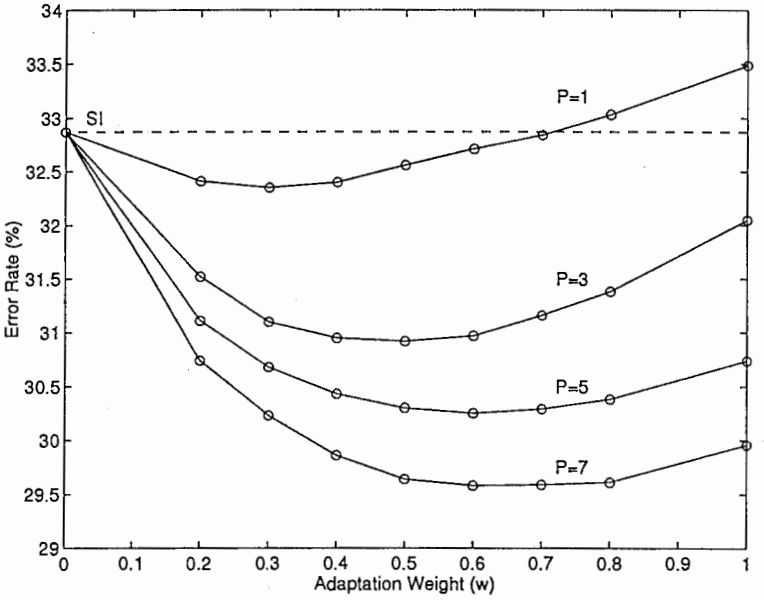

図 5: Phonetic recognition performance of PVP using $K = 2$ as the smoothing weight w and the number of adaptation phrases P are varied.

5.5 PVP vs. MAP vs. VFS

Figure 6 shows the performance of PVP using optimized smoothing weights in comparison to the standard MAP algorithm using optimized values of τ. As can be seen PVP performs marginally better than MAP, achieving approximately the same reduction in error as MAP with one fewer adaptation phrase than MAP. Experiments to compare PVP to VFS were also conducted. For this task, VFS performed worse than standard MAP for all tested cases. This indicates that for this small task the VFS smoothing process harms the MAP predicted values.

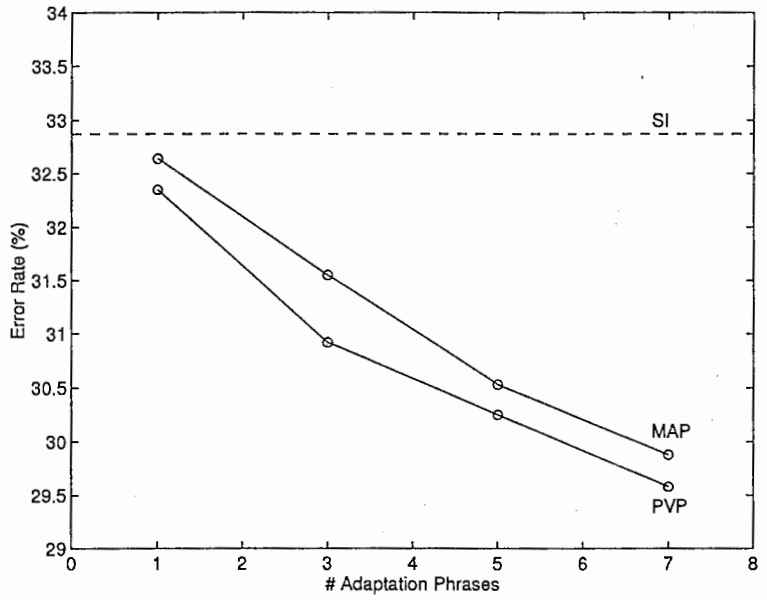

図 6: Phonetic recognition performance of PVP with optimized smoothing weights vs. MAP with optimized values for τ as the number of adaptation phrases is varied.

6 Conclusion

The results from the experiments presented in this paper allow two primary conclusions to be drawn. First, the use of *a priori* information about within speaker correlations across states can be useful for improving speaker adaptation performance. However, this information is difficult to capture efficiently and care must be taken to insure the proper balance between complexity and trainability. The tremendous reduction in the parameter set size that must be undertaken in order for PVP to be effective is evidence for this belief. By reducing the number of correlation matrices used per class down to $K = 2$ much of the useful correlation information from other classes has been removed. However this reduction in parameter size is necessary to insure that the estimation noise introduced from the large number of parameters that are potentially undertrained is kept in check. In order for methods such as PVP to work more effectively, means of capturing the most relevant correlation information in an efficient representation must be developed. Blind selection of the correlation matrices and parameters to be used will not achieve optimal results.

The second conclusion that can be made is that VFS smoothing does not work well across states that do not belong to the same phoneme. VFS has been shown to be effective for an HMM system with 200 states with a mixture of 5 Gaussian per state. In this system, VFS smoothing of the MAP estimated transfer vectors of the 6 nearest neighbor distributions allows for considerable improvement over standard MAP adaptation techniques. However, within this system the smoothing is primarily performed using distributions from the same mixture or distributions from the mixture of the nearest adjacent state, which is often just a different context dependent state for the same phoneme. Thus, the linear interpolation is most often performed on distributions stemming from the same phoneme. However, with only 50 states and 1 Gaussian per state, smoothing is usually performed across states belonging to different phonemes. Under these conditions, VFS does not perform well. Thus, we must conclude that VFS primarily helps in smoothing the estimates made across distributions representing the same phoneme but does not help in predicting the transfer vectors of states from one phoneme based on observations from different phonemes.

参考文献

[1] M. J. Lasry and R. M. Stern. *A posteriori* estimation of correlated jointly Gaussian mean vectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(4):530–535, July 1984.

[2] K. Ohkura, M. Sugiyama, and S. Sagiyama. Speaker adaptation based on transfer vector field smoothing with continuous mixture density HMMs. In *Proceedings of the 1992 International Conference on Spoken Language Processing*, pages 369–372, 1992.

[3] R. M Stern and M. J. Lasry. Dynamic speaker adaptation for feature-based isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(6):751–763, June 1987.

[4] M. Tonomura, T. Kosaka, and S. Matsunaga. Speaker adaptation based on transfer vector field smoothing using maximum *a posteriori* probability estimation. In *Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing*, pages 681–691. IEEE, 1995.

[5] G. Zavaliagkos, R. Schwartz, and J. Makhoul. Batch, incremental, and instantaneous adaptation techniques for speech recognition. In *Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing*, pages 676–679. IEEE, 1995.