Odyssey 2012
The Speaker and Language Recognition Workshop
25-28 June 2012, Singapore

A Small Footprint i-Vector Extractor

Patrick Kenny

Centre de recherche informatique de Montréal (CRIM)
Patrick.Kenny@crim.ca

Abstract

Both the memory and computational requirements of algorithms traditionally used to extract i-vectors at run time and to train i-vector extractors off-line scale quadratically in the i-vector dimensionality. We describe a variational Bayes algorithm for calculating i-vectors *exactly* which converges in a few iterations and whose computational and memory requirements scale linearly rather than quadratically. For typical i-vector dimensionalities, the computational requirements are slightly greater than those of the traditional algorithm. The run time memory requirement is scarcely greater than that needed to store the eigenvoice basis. Because it is an exact method, the variational Bayes algorithm enables the construction of i-vector extractors of much higher dimensionality than has previously been envisaged. We show that modest gains in speaker verification accuracy (as measured by the 2010 NIST detection cost function) can be achieved using high dimensional i-vectors.

1. Introduction

An important recent advance in speaker recognition is the discovery that speech signals of arbitrary duration can be effectively represented by i-vectors of relatively low dimension (with this dimension being independent of the utterance duration) [1]. I-vectors have worked equally well in language recognition [2, 3] and both i-vectors and speaker factors extracted from segments of very short duration (on the order of 1 second) have been successfully used in diarizing telephone conversations [4, 5, 6]. (Speaker factors are extracted in the same way as i-vectors. They differ in the way the data used to estimate the eigenvoice basis is organized.)

I-vectors are so easy to work with that they are now the primary representation used in many state of the art speaker recognition systems. By banishing the time dimension altogether, the i-vector representation enables the speaker recognition problem to be cast as a traditional biometric pattern recognition problem like face recognition. This allows well established techniques such as cosine distance scoring, Linear Discriminant Analysis and Probabilistic Linear Discriminant Analysis (PLDA) to be applied. The advantage of a low, fixed dimensional feature representation is particularly apparent in the case of PLDA as this can be regarded as a simplified version of Joint Factor Analysis (JFA) which results when each utterance is represented by single feature vector (rather than by a sequence of cepstral vectors of random duration, as is traditional in speech processing).

The i-vector representation of speech utterances can be viewed as a type of principal components analysis of utterances of arbitrary duration, based on the assumption that each utterance can be modeled by a Gaussian Mixture Model (GMM) and applying probabilistic principal components analysis to the GMM supervectors. Thus the basic assumption is that all utterance supervectors are confined to a low dimensional subspace of the GMM supervector space so that each supervector is specified by a small number of coordinates. These coordinates can be thought of as representing physical quantities which are constant for a given utterance (such as vocal tract length, room impulse response etc.) but which differ from one utterance to another. The i-vector representation of the utterance is defined by these coordinates. The standard algorithm for extracting i-vectors is eigenvoice maximum *a posteriori* (MAP) estimation (Proposition 1 of [7]) and the primary training algorithm used in building an i-vector extractor is the eigenvoice estimation algorithm given in Proposition 3 of [7] (applied in such a way that each utterance is treated as coming from a different speaker).

Both the computational and memory requirements of these algorithms scale *quadratically* in the i-vector dimensionality. This accounts for the fact that, although principal components analyzers of several thousand dimensions are commonly used in other fields, there are to our knowledge no instances in the current speaker recognition literature of i-vector extractors of dimension greater than 800. For example an i-vector extractor of dimension 1000 (with a standard configuration of 2048 Gaussians and 60 dimensional acoustic features) requires 8.8 Gb of storage (in double precision) at run time and twice that amount of memory is needed to train it.

In this paper, we will show how to minimize the memory requirements of i-vector extraction (both at run time and during training) using an iterative variational Bayes algorithm to perform the eigenvoice MAP computation. The algorithm converges very quickly (3 variational Bayes iterations are typical) and no cost in speaker recognition accuracy is incurred. A 1000 dimensional i-vector extractor can be accommodated in less than 1 Gb at run time and the computational overhead is quite modest: the CPU time required to extract a 1000 dimensional i-vector from an utterance is on the order of 1 second, assuming that the Baum-Welch statistics for the utterance are given.

A key aspect of the variational Bayes algorithm is that both the computational and memory requirements scale *linearly* rather than quadratically in the i-vector dimensionality. This makes it possible to explore the question of whether improvements in speaker recognition accuracy can be obtained using very high dimensional i-vector representations. (It is well known that approximate methods can be used to extract i-vectors at run time without seriously compromising speaker recognition accuracy but exact computations appear to be needed for training i-vector extractors [8]. This was our principal motivation for developing the variational Bayes method.) We have experimented with i-vectors of dimension as high as 1600 and we will show that minor improvements in the value of the normalized 2010 NIST detection cost function can be obtained using very high dimensional i-vectors.

2. Review of i-vectors

2.1. Notation

The probabilistic model underlying eigenvoice MAP is an extension of probabilistic principal components analysis [9]. We assume that we are given a universal background model (UBM) with C mixture components indexed by $c = 1, \ldots, C$. For each mixture component c we denote by w_c, m_c and Σ_c the corresponding mixture weight, mean vector and covariance matrix. We denote by F the dimension of the acoustic feature vectors.

To account for inter-utterance variability, we associate with each utterance an $R \times 1$ vector $\boldsymbol{y}$ (the i-vector) and with each mixture component c an $F \times R$ matrix V_c. For the given utterance, the acoustic feature vectors associated with the mixture component are supposed to be distributed with mean μ_c and covariance matrix Σ_c where

$$\mu_c \quad = \quad m_c + V_c \boldsymbol{y}.$$

Thus, if we are given an utterance represented as a sequence of frames $X_1, \ldots, X_T$ and the aliment of frames with mixture components is given, the likelihood of the utterance is

$$\sum_c \Big(N_c \ln \frac{1}{(2\pi)^{F/2} |\Sigma_c|^{1/2}} - \frac{1}{2} \sum_t (X_t - V_c \boldsymbol{y} - m_c)^* \Sigma_c^{-1} (X_t - V_c \boldsymbol{y} - m_c) \Big)$$

where the sum over c extends over all mixture components; the sum over t extends over all frames aligned with the mixture component c; and N_c is the number of such frames. This expression can be evaluated in terms of the first and second order statistics for each mixture component, namely

$$\begin{aligned} F_c &= \sum_t X_t \\ S_c &= \sum_t X_t X_t^*. \end{aligned}$$

Since the alignment of frames with mixture components is *not* in fact given, we use the Baum-Welch statistics instead. These are defined by

$$\begin{aligned} N_c &= \sum_t \gamma_t(c) \\ F_c &= \sum_t \gamma_t(c) X_t \\ S_c &= \sum_t \gamma_t(c) X_t X_t^* \end{aligned}$$

where, for each time t, $\gamma_t(c)$ is the posterior probability that X_t is generated by the mixture component c, calculated with the UBM. (This choice is intuitively natural and it can be motivated by variational Bayes [10].)

In [7], the only roles played by the second order Baum-Welch statistics are in the calculation of the likelihood function (Proposition 2) and in the estimation of the covariance matrices Σ_c (Proposition 3). In practice, for each mixture component c, m_c and Σ_c can be copied from the UBM and only the matrix V_c needs to be estimated from the training data. Furthermore, the contribution of the second order statistics can be dropped from the expression for the likelihood function without compromising its usefulness. Thus the second order statistics do not play an essential role in most implementations.

2.2. i-vector extraction

Proposition 1 of [7] shows how to calculate the posterior distribution of $\boldsymbol{y}$ for a given utterance on the assumption that the prior is standard normal. (There is no gain in generality by assuming a non-standard normal prior.) The posterior covariance matrix $\mathrm{Cov}(\boldsymbol{y}, \boldsymbol{y})$ and mean vector $\langle \boldsymbol{y} \rangle$ are given by

$$\begin{aligned} \mathrm{Cov}(\boldsymbol{y}, \boldsymbol{y}) &= \left(I + \sum_c N_c V_c^* \Sigma_c^{-1} V_c \right)^{-1} \\ \langle \boldsymbol{y} \rangle &= \mathrm{Cov}(\boldsymbol{y}, \boldsymbol{y}) \sum_c V_c^* \Sigma_c^{-1} F_c. \end{aligned} \tag{1}$$

The usual procedure to alleviate the computational burden is to store the matrices $V_c^* \Sigma_c^{-1} V_c$ and, for each utterance s, use these matrices together with the 0 order Baum-Welch statistics to evaluate the precision matrix

$$I + \sum_c N_c(s) V_c^* \Sigma_c^{-1} V_c$$

for the given utterance. Because of symmetry, only $R(R+1)/2$ real numbers (i.e. the upper triangular part of $V_c^* \Sigma_c^{-1} V_c$) need to be stored for each mixture component. Of course it would be possible to economize on memory by calculating the precision matrix for each utterance from scratch but this is not done in practice as the computational burden would be excessive.

2.3. Whitening the Baum-Welch statistics

A simplification which can be applied if the mean vectors and covariance matrices m_c and Σ_c are taken as given (rather than estimated in the course of training the i-vector extractor) is that the first order statistics can be pre-whitened by subjecting them to the transformation

$$F_c \quad \leftarrow \quad L_c^{-1}(F_c - N_c m_c)$$

where

$$L_c L_c^* = \Sigma_c$$

is the Cholesky decomposition of Σ_c. (Of course changing the Baum-Welch statistics in this way will change the the estimates of the matrices V_c produced by the eigenvoice training algorithm summarized below. But these changes cancel each other out in the sense that, for each utterance, the posterior distribution of the hidden variables $\boldsymbol{y}$ remains unchanged. Thus the point estimate of the i-vector — that is, the mean of the posterior — remains unchanged which is all that we require.)

Performing this transformation enables us to take $m_c = 0$ and $\Sigma_c = I$ in all of the equations in [7]. This simplifies the implementation and, more importantly, it facilitates building i-vector extractors which do not have to distinguish between full covariance and diagonal covariance UBMs. (It was shown in [11] that full covariance UBMs outperform diagonal covariance UBMs in speaker recognition.) We will assume henceforth that the Baum-Welch statistics have been whitened.

2.4. Training the ivector extractor

As for estimating the matrices V_c, suppose we have a training set where for each utterance s, the first and second order moments $\langle \boldsymbol{y}(s) \rangle$ and $\langle \boldsymbol{y}(s) \boldsymbol{y}^*(s) \rangle$ have been calculated. The maximum likelihood update formula for V_c is

$$V_c \quad = \quad \left(\sum_s \langle \boldsymbol{y}(s) \rangle F_c^*(s) \right) \left(\sum_s N_c(s) \langle \boldsymbol{y}(s) \boldsymbol{y}^*(s) \rangle \right)^{-1}$$

where the sums over s extend over all utterances in the training set, and for each utterance s, $N_c(s)$ and $F_c(s)$ are the Baum-Welch statistics of order 0 and 1 for mixture component c. Calculating the first and second moments is just a matter of evaluating the posterior of $\boldsymbol{y}(s)$. For example,

$$\langle \boldsymbol{y}(s)\boldsymbol{y}^*(s)\rangle \quad = \quad \mathrm{Cov}\,(\boldsymbol{y}(s),\boldsymbol{y}(s)) + \langle \boldsymbol{y}(s)\rangle\langle \boldsymbol{y}^*(s)\rangle.$$

As for minimum divergence estimation, the idea is to modify the matrices V_c in such a way as to force the empirical distribution of the i-vectors to conform to the standard normal prior [12, 13]. Let LL^* be the Cholesky decomposition of the matrix

$$\frac{1}{S}\sum_s \langle \boldsymbol{y}(s)\boldsymbol{y}^*(s)\rangle$$

where S is the number of training utterances and the sum extends over all utterances in the training set. The transformations

$$\begin{aligned} V_c &\leftarrow V_cL \\ \boldsymbol{y}(s) &\leftarrow L^{-1}\boldsymbol{y}(s) \end{aligned}$$

have the desired effect.

Both maximum likelihood and minimum divergence estimation increase the likelihood of the training set where the likelihood is evaluated as in Proposition 2 of [7]. Note that, just as for run-time i-vector extraction, calculating i-vector posteriors is the principal computation for both maximum likelihood and minimum divergence training.

3. The variational Bayes algorithm

We will show how the memory requirements of the posterior calculation can be substantially reduced at the cost of a modest computational overhead by working with a basis of the i-vector space with respect to which the i-vector posterior covariance matrices are approximately diagonal, so that the i-vector components are approximately statistically independent in the posterior.

The variational Bayes (VB) method is a standard way of enforcing this type of posterior independence assumption. In the case at hand, variational Bayes produces an approximation to the true posterior which is of very high quality in the sense that, at convergence, the mean of the posterior distribution — that is, the point estimate of the i-vector — is calculated *exactly*. The only inaccuracy is in the posterior covariance matrix which is only approximately diagonal. These posterior covariances are used in training the i-vector extractor (not at run time) but the effect of the diagonal approximation is minimal so that it turns out that using the VB method in training i-vector extractors leads to *no degradation in speaker recognition accuracy*.

If the VB algorithm is initialized properly, very few iterations (typically 3, independently of the i-vector dimension) are needed to obtain accurate i-vector estimates. Because the off-diagonal elements of the posterior covariance matrix are ignored, it follows that *the computational and memory requirements of the VB method scale linearly rather than quadratically in the i-vector dimension.*

3.1. VB updates

Suppose we are given an utterance represented as a sequence of acoustic feature vectors $X_1,\dots,X_T$ or $\boldsymbol{X}$ for short. We suppose that the prior distribution of $\boldsymbol{y}$ is standard normal. Assume provisionally that it is reasonable to impose diagonal constraints on the posterior covariance matrix of $\boldsymbol{y}$ given $\boldsymbol{X}$ so that we can write

$$\begin{aligned} \boldsymbol{y} &= (y^1,\dots,y^R) \\ Q(\boldsymbol{y}) &= Q(y^1)\dots Q(y^R). \end{aligned}$$

where $Q(\boldsymbol{y})$ approximates the true posterior $P(\boldsymbol{y}|\boldsymbol{X})$. We will return to the question of why this assumption is reasonable in the next section. Meantime, we derive a variational Bayes algorithm to calculate $Q(\boldsymbol{y})$. We introduce the following notation. For $r=1,\dots,R$, we denote the rth column of V_c by V_c^r so that

$$V_c\boldsymbol{y} \quad = \quad \sum_{r=1}^R V_c^r y^r$$

and

$$\mathrm{diag}\,(V_c^*V_c) \quad = \quad \begin{pmatrix} V_c^{1*}V_c^1 & & \\ & \ddots & \\ & & V_c^{R*}V_c^R \end{pmatrix}.$$

The memory overhead of the VB algorithm is the cost of storing this diagonal matrix for each mixture component c, rather than (the upper triangle of) V_cV_c as required by the standard posterior calculation outlined in Section 2.2.

Following the standard procedure given in [9] the VB update for $Q(y^r)$ is

$$\ln Q(y^r) = E_{\boldsymbol{y}\setminus y^r}\left[\ln\frac{P(\boldsymbol{y},\boldsymbol{X})}{Q(\boldsymbol{y})}\right] + \text{constant}.$$

So using $\equiv$ to indicate equality up to an additive constant, we have

$$\begin{aligned} &\ln Q(y^r) \\ &\equiv \quad -\frac{1}{2}\sum_c E_{\boldsymbol{y}\setminus y^r}\left[\sum_t (X_t - V_c\boldsymbol{y})^*(X_t - V_c\boldsymbol{y})\right] - \frac{1}{2}(y^r)^2 \\ &\equiv \quad -\frac{1}{2}\left(1 + N_cV_c^{r*}V_c^r\right)(y^r)^2 \\ &\qquad + \sum_c V_c^{r*}\Big(F_c - N_c\sum_{r'\neq r}\langle y^{r'}\rangle V_c^{r'}\Big)y^r. \end{aligned}$$

Letting

$$L^r \quad = \quad 1 + \sum_c N_cV_c^{r*}V_c^r,$$

we can read off the posterior expectation and variance of y^r from this expression:

$$\begin{aligned} \langle y^r\rangle &= \frac{1}{L^r}\sum_c V_c^{r*}\Big(F_c - N_c\sum_{r'\neq r}\langle y^{r'}\rangle V_c^{r'}\Big) \\ \mathrm{Var}(y^r) &= \frac{1}{L^r}. \end{aligned}$$

A complete VB update iteration consists in applying this for $r=1,\dots,R$. Readers familiar with the Jacobi method in numerical linear algebra will recognize that successive VB iterations implement the Jacobi method for solving the linear system (1). It is well known (and easy to verify) that, if the Jacobi method converges, then it converges to the solution of the linear system. Convergence in this case is guaranteed by variational

Bayes [9]. Thus, if is run to convergence, the VB method calculates i-vectors exactly.

For efficient implementation, we can use the following reformulation. Set $R_c = F_c - N_c V_c \langle \boldsymbol{y} \rangle$ for each mixture component c (R for remainder). The VB update of $\langle \boldsymbol{y} \rangle$ consists in performing the following operations for $r = 1, \ldots, R$:

1. $R_c \leftarrow R_c + N_c \langle y^r \rangle V_c^r \;\; (c = 1, \ldots, C)$
2. $\langle y^r \rangle = \frac{1}{L^r} \sum_c V_c^{r*} R_c$
3. $R_c \leftarrow R_c - N_c \langle y^r \rangle V_c^r \;\; (c = 1, \ldots, C)$

More efficiently (since $\operatorname{diag}(V_c^* V_c)$ has been precomputed), 1) and 2) can be combined as

$$\begin{aligned} \langle y^r \rangle_{\text{new}} &= \frac{1}{L^r} \sum_c V_c^{r*} \left(R_c + N_c \langle y^r \rangle_{\text{old}} V_c^r \right) \\ &= \frac{1}{L^r} \sum_c V_c^{r*} R_c + \langle y^r \rangle_{\text{old}} \frac{1}{L^r} \sum_c N_c V_c^{r*} V_c^r \\ &= \frac{1}{L^r} \sum_c V_c^{r*} R_c + \left(1 - \frac{1}{L^r} \right) \langle y^r \rangle_{\text{old}} \end{aligned}$$

so that the VB update of $\langle \boldsymbol{y} \rangle$ reduces to performing the following operations for $r = 1, \ldots, R$:

1. $\langle y^r \rangle_{\text{new}} = \frac{1}{L^r} \sum_c V_c^{r*} R_c + \left(1 - \frac{1}{L^r} \right) \langle y^r \rangle_{\text{old}}$
2. $R_c \leftarrow R_c - N_c \left(\langle y^r \rangle_{\text{new}} - \langle y^r \rangle_{\text{old}} \right) V^r \quad (c = 1, \ldots, C)$.

3.2. VB Initialization

The VB algorithm is guaranteed to calculate i-vectors exactly but a good initialization is needed to ensure that it does so quickly.

We are free to postmultiply the matrices V_c by any orthogonal matrix without affecting the assumption that $\boldsymbol{y}$ has a standard normal prior distribution; in particular we can choose a basis of the i-vector space such that the matrix

$$\sum_c w_c V_c^* V_c$$

is diagonal where, for each mixture component c, w_c is the corresponding mixture weight (i.e. prior probability). Since, for a given utterance of sufficiently long duration,

$$N_c \approx N w_c$$

it follows that the posterior covariance matrix for an utterance, namely $\operatorname{Cov}(\boldsymbol{y}, \boldsymbol{y})$ in (1), is approximately diagonal (as we mentioned in Section 3.1). Note that the quality of the approximation may degrade in the case of very short utterances (for which it may not be the case that $N_c \approx N w_c$ for each mixture component component). Thus a reasonable initial estimate of $\langle \boldsymbol{y} \rangle$ for VB is the approximate solution of (1) obtained by ignoring the off-diagonal elements in $\operatorname{Cov}(\boldsymbol{y}, \boldsymbol{y})$. (Solving a system of equations with a diagonal coefficient matrix is computationally trivial.) Unlike the approximation used in [8], this is only an initial estimate chosen so as to ensure rapid convergence of the VB algorithm. Note that in [8] the approximation is only for extracting i-vectors at run time; it turns out to be too crude to use in training i-vector extractors. On the other hand, since the variational Bayes approach to calculating i-vectors is exact, it can be used in off-line training of i-vector extractors. This is the reason why we are able to experiment with i-vectors of very high dimensionality.

3.3. The variational lower bound

The variational Bayes updates are guaranteed to increase the variational lower bound $\mathcal{L}$ on $\ln P(\boldsymbol{X})$ defined by

$$\mathcal{L} = E\left[\ln \frac{P(\boldsymbol{y}, \boldsymbol{X})}{Q(\boldsymbol{y})} \right]$$

where the expectation is taken with respect to $Q(\boldsymbol{y})$. Ignoring the contribution of the second order statistics (which does not change from one VB iteration to the next), the variational lower bound can be expressed in terms of the posterior mean and covariance as follows

$$\begin{aligned} \mathcal{L} &= \sum_c (V_c \langle \boldsymbol{y} \rangle)^* F_c - \frac{1}{2} \sum_c N_c (V_c \langle \boldsymbol{y} \rangle)^* V_c \langle \boldsymbol{y} \rangle \\ &\quad - \frac{1}{2} \sum_c \operatorname{tr}\left(N_c V_c^* V_c \operatorname{Cov}(\boldsymbol{y}, \boldsymbol{y}) \right) - D\left(Q(\boldsymbol{y}) \| P(\boldsymbol{y}) \right) \end{aligned}$$

where $D\left(Q(\boldsymbol{y}) \| P(\boldsymbol{y}) \right)$ is the Kullback-Leibler divergence between the posterior $Q(\boldsymbol{y})$ and the standard normal prior $P(\boldsymbol{y})$ which (by the formula for the divergence of two multivariate Gaussians [13]) is given by

$$\begin{aligned} D\left(Q(\boldsymbol{y}) \| P(\boldsymbol{y}) \right) &= -\frac{R}{2} - \frac{1}{2} \ln |\operatorname{Cov}(\boldsymbol{y}, \boldsymbol{y})| \\ &\quad + \frac{1}{2} \operatorname{tr}\left([\operatorname{Cov}(\boldsymbol{y}, \boldsymbol{y}) + \langle \boldsymbol{y} \rangle \langle \boldsymbol{y}^* \rangle] \right). \end{aligned}$$

Evaluating the variational lower bound on each VB update turns out to be rather expensive so in practice it is more useful for troubleshooting than for monitoring convergence of the VB algorithm at run time. (A simple, effective run-time convergence criterion is the Euclidean norm of the difference between successive i-vector estimates $\| \boldsymbol{y}_{\text{new}} - \boldsymbol{y}_{\text{old}} \|$.)

In constructing an i-vector extractor, the usual criterion used to monitor convergence of the maximum likelihood and minimum divergence training algorithms is the exact likelihood function described in Proposition 2 of [7] whose evaluation requires computing exact posterior covariances. If the posteriors are calculated with the VB algorithm, the appropriate criterion to use is the aggregate variational lower bound calculated by summing the variational lower bound over all training utterances. If posteriors are evaluated with the VB method, this criterion is guaranteed to increase from one training iteration to the next (both for maximum likelihood training and minimum divergence training) so it is useful for troubleshooting.

4. Experimental Results

4.1. Testbed

We report the results of experiments conducted on the det 2 trials (normal vocal effort telephone speech) in the female portion of the extended core condition of the NIST 2010 speaker recognition evaluation (thus using a larger set of trials than was provided for in the original evaluation plan[1]). Results are reported using the following metrics: the equal error rate (EER), the normalized detection cost function used in evaluations prior to 2010 (2008 NDCF) and the normalized detection cost function defined for the 2010 evaluation (2010 NDCF) which severely penalizes false alarms. The purpose of the experiments was to compare the performance of the VB i-vector extractor with that of our earlier JFA-based implementation and to investigate the

[1] http://www.itl.nist.gov/iad/mig/tests/spk/2010/NIST_SRE10_evalplan.r6.pdf

question of whether improvements in speaker verification accuracy could be achieved with very high dimensional i-vectors.

Speaker verification was carried out with heavy-tailed PLDA, using the front end and data sets for training the UBM, the i-vector extractors and the PLDA classifiers described in [13]. In comparing the VB i-vector extractor with the JFA-based implementation, we used the in house CRIM voice activity detector (VAD) in extracting Baum-Welch statistics. However we learned in the course of the Bosaris workshop at BUT in 2010 that the CRIM voice activity detector did not perform as well as the celebrated BUT Hungarian phoneme recognizer and BUT kindly made their transcripts available to us for the experiments with very high dimensional i-vectors.

4.2. Accuracy of VB

We evaluated the accuracy of the variational Bayes method by comparing it with our original i-vector implementation, where the i-vector extractor was built using executables written for Joint Factor Analysis. (The scripts were modified in such a way as to ignore the speaker labels attached to recordings in the Switchboard and Mixer training corpora.) We used a standard UBM configuration (2048 diagonal Gaussians, 60 dimensional acoustic feature vectors) to extract Baum-Welch statistics. However, in our JFA implementation, the Baum-Welch statistics were not whitened, the means and covariances associated with the various mixture components were re-estimated in the course of factor analysis training rather than copied form the UBM (and only UBMs with diagonal covariance matrices were supported).

To test the accuracy of the variational Bayes method we used it in conjunction with the diagonal covariance UBM both to train an i-vector extractor and to calculate the i-vectors at run time. Our implementation in this case used whitened Baum-Welch statistics so the means and covariances associated with the various mixture components were *not* re-estimated in the course of training the i-vector extractor. We used 5 iterations of variational Bayes to extract i-vectors.

The i-vector dimensionality was 400 in both cases, reduced to 100 by linear discriminant analysis as in [1, 11] and heavy-tailed PLDA classifiers [13] were used for verification. The results were essentially identical: an equal error rate of 3.1% for the JFA implementation versus 3.0% for VB and a detection cost of 0.50 for JFA versus 0.49 for VB (where the detection cost was measured with the 2010 NIST cost function).

The slight edge observed for the VB method may seem surprising. It appears to be attributable to the fact that the variances were copied from the UBM in this case but not in the case of the JFA-based implementation. The effect of this copying is to overestimate the variances in the i-vector extractor by about 5%. (Copying results in overestimates since the UBM variances are estimated in a way which takes no account of inter-utterance variability.) It is well known that overestimating variances often proves to be helpful in speech modeling.

4.3. Efficiency of VB

We evaluated the efficiency of the VB method by performing a comparison with the standard approach summarized in (1), using whitened Baum-Welch statistics in both cases. On a 2.40 GHz Intel Xeon CPU, the time taken to extract an i-vector with the standard approach was about 0.5 seconds. Almost all of the time was spent in BLAS routines (in particular 75% of the computational burden is spent accumulating the covariance matrix in (1)). An estimate of about 0.25 seconds is given [8] which appears to indicate that compiler optimization might be useful.

For the VB approach we performed 5 VB updates at run time. Under these conditions the time taken to extract an i-vector was 0.9 seconds. Thus, as expected, the VB method is slower (by about a factor of 2 in the case of 400 dimensional i-vectors) but still quite quick compared with the cost of extracting Baum-Welch statistics using a large UBM. In working with higher dimensional i-vectors, we always found that 3–5 variational Bayes iterations were sufficient. (The exact number of iterations performed was determined by the Euclidean norm stopping criterion mentioned in Section 3.3). It follows that, like the memory requirements, the computational overhead of the VB method scales *linearly* rather than quadratically in the i-vector dimension. So the computational advantage of the standard implementation relative to the VB method actually *decreases* as the i-vector dimension increases.

4.4. High dimensional i-vectors

Since the VB method enables very high dimensional i-vector extractors to be trained with relatively modest computational resources, we conducted some experiments to see if any gains in accuracy could be achieved by increasing the i-vector dimensionality. We used the same 2048 component diagonal UBM as in the previous sections but we replaced the CRIM VAD with the BUT VAD which accounts for the lower error rates. In every case we reduced the number of dimensions to 100 by linear discriminant analysis (LDA). Table 1 shows the results of increasing the i-vector dimension from 400 to 1600 in steps of 400. The equal error rates degrade, but there is a minor improvement in the 2010 NDCF.

Table 1: *EER / 2010 NDCF female extended core condition. 100 dimensional LDA. Diagonal UBM, 2048 Gaussians.*

i-vector	EER	2008 NDCF	2010 NDCF
400	**2.5%**	0.13	0.45
800	2.7%	0.13	0.44
1200	2.7%	0.13	**0.42**
1600	2.8%	0.14	0.43

5. Conclusion

We have shown how a variational Bayes approach enables exact i-vector extraction to performed in such a way that computational and memory requirements scale linearly rather than quadratically in the i-vector dimensionality. Because it is an exact method, the variational Bayes algorithm can be used in training i-vector extractors as well as at run-time. This makes it possible to experiment with i-vectors of much higher dimension than has previously been envisaged, yielding modest improvements in speaker verification accuracy as measured by the NIST 2010 detection cost function.

Acknowledgements

We would like to thank Brno University of Technology for hosting the 2010 Bosaris workshop where this work was begun. Special thanks to Ondrej Glembek who suggested the initialization in Section 3.2.

6. References

[1] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 4, pp. 788–798, May 2011.

[2] N. Dehak, P. Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language recognition via ivectors and dimensionality reduction," in *Proc. Interspeech*, Florence, Aug. 2011.

[3] G. Martnez, O. Plchot, L. Burget, O. Glembek, and P. Matejka, "Language recognition in ivectors space," in *Proc. Interspeech*, Florence, Aug. 2011.

[4] F. Castaldo, D. Colibro, E. Dalmasso, P. Laface, and C. Vair, "Stream-based speaker segmentation using speaker factors and eigenvoices," in *Proc. ICASSP*, Las Vegas, Nevada, Mar. 2008, pp. 4133 – 4136.

[5] C. Vaquero, A. Ortega, and E. Lleida, "Intra-session variability compensation and a hypothesis generation and selection strategy for speaker segmentation," in *Proc. ICASSP*, 2011, pp. 4532–4535.

[6] S. Shum, N. Dehak, E. Chuangsuwanich, D. Reynolds, and J. Glass, "Exploiting intra-conversation variability for speaker diarization," in *Proc. Interspeech*, Florence, Aug. 2011.

[7] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE Trans. Speech Audio Processing*, vol. 13, no. 3, pp. 345–359, May 2005.

[8] O. Glembek, L. Burget, P. Matejka, M. Karafiat, and P. Kenny, "Simplification and optimization of i-vector extraction," in *Proceedings ICASSP*, 2011.

[9] C. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer Science+Business Media, LLC, 2006.

[10] X. Zhao, Y. Dong, J. Zhao, L. Lu, J. Liu, and H. Wang, "Variational Bayesian joint factor analysis for speaker verification," in *Proc. ICASSP 2009*, Taipei, Taiwan, Apr. 2009.

[11] P. Matejka, O. Glembek, F. Castaldo, J. Alam, O. Plchot, P. Kenny, L. Burget, and J. Cernocky, "Full-covariance ubm and heavy-tailed plda in i-vector speaker verification," in *Proceedings ICASSP*, 2011.

[12] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel, "A study of inter-speaker variability in speaker verification," *IEEE Trans. Audio, Speech and Lang. Process.*, vol. 16, no. 5, pp. 980–988, July 2008. [Online]. Available: http://www.crim.ca/perso/patrick.kenny

[13] P. Kenny, "Bayesian speaker verification with heavy tailed priors," in *Proc. Odyssey 2010: The speaker and Language Recognition Workshop*, Brno, Czech Rebublic, June 2010.