Supervised I-vector Modeling for Language and Accent Recognition

Shreyas Ramoji, Sriram Ganapathy

Learning and Extraction of Acoustic Patterns (LEAP) Lab, Department of Electrical Engineering, Indian Institute of Science, Bengaluru

Abstract

The conventional i-vector approach to speaker and language recognition constitutes an unsupervised learning paradigm where a variable length speech utterance is converted into a fixed dimensional feature vector (termed as i-vector). The i-vector approach belongs to the broader family of factor analysis models where the utterance level adapted means of a Gaussian Mixture Model - Universal Background Model (GMM-UBM) are assumed to lie in a low rank subspace. The latent variables in the low rank model are assumed to have a standard Gaussian prior distribution. In this paper, we rework the theory of i-vector modeling in a supervised framework where the class labels (like language or accent) of the speech recordings are introduced directly into the i-vector model using a mixture Gaussian prior where each mixture component is associated with a class label. We provide the mathematical formulation for minimum mean squared error estimate (MMSE) of the supervised i-vector (s-vector) model. A detailed analysis of the s-vector model is given and this is contrasted with the traditional i-vector framework. The proposed model is used for language recognition tasks using the NIST Language Recognition Evaluation (LRE) 2017 dataset as well as an accent recognition task using the Mozilla common voices dataset. In these experiments, the s-vector model provides significant improvements over the conventional i-vector model (relative improvements of up to 24% for LRE task in terms of primary detection cost metric).

Keywords: Unsupervised i-vector, s-vector, minimum-mean square error (MMSE) estimate, language recognition.

1. Introduction

The problem of language (accent) recognition involves the development of algorithms that automatically infer the language (accent) from a speech recording. Language and accent recognition has important applications in call centers, helplines, voice assistants, robotics and also in security and defence applications. Over the last three decades, various approaches like phonotactic, acoustic modeling using statistical methods and recent approaches involving neural networks have been explored for the development of language (accent) recognition systems. Factors such as the presence of multiple speakers, content based variability in short duration speech, noise etc severely degrade the performance of language (accent) recognition systems. The National Institute of Standards and Technology (NIST) has been organizing a series of Language Recognition Evaluations (LRE) since 1996 to benchmark the performances of LID systems and to highlight the challenging scenarios where the systems typically fail. The last two cycles (LRE 2015, 2017) focused on distinguishing between closely related languages and dialects, which is a challenging problem.

Traditionally, phoneme recognition followed by language modeling (PRLM) was one of the popular methods for automatic LID task [1, 2]. This approach uses a multilingual phoneme recognizer to generate phoneme sequences which are converted to language model (n-gram) features for the LID classifier. The success of this approach is dependent on the performance of the phoneme decoder. For relatively clean data with linguistic resources, the PRLM method provides good performance comparable to acoustic systems [3]. However, most of the successful LID systems developed in the last decade (on noisy data) are devoid of phoneme recognition engines.

Regarding the features for language recognition, while traditional systems use mel frequency cepstral coefficients (MFCC) [4] or shifted delta coefficient (SDC) [5] features, the deep neural network (DNN) based posterior features were attempted recently for LID [6]. The Tandem features have shown promising results for noisy language recognition as well [7]. The use of bottleneck features derived from a speech recognition acoustic model has also shown to benefit language recognition [8, 9, 10]. In this paper, we use acoustic features based on bottleneck representations from a deep neural network trained for speech recognition using Switchboard English corpus.

In the last decade, the most popular approach to language (accent) and speaker recognition consists of modeling a database of speech recordings (in

the form of a sequence of short-term feature vectors) with a Gaussian Mixture Model - Universal Background Model (GMM-UBM) [11, 12]. The initial approaches for speaker recognition using log-likelihood scores were replaced with the factor analysis models [13] where the adapted Gaussian mean components (spliced as a single high dimensional vector called the supervector) are expressed as a sum of speaker and session factors. The parameters in this model were derived using a maximum likelihood (ML) framework with an iterative expectation maximization (EM) approach. The approach of joint factor analysis (JFA) was further simplified by total variability modeling (i-vector modeling) where all variabilities were captured by a single fixed dimensional latent vector [14]. With a prior of standard normal distribution (having zero mean and identity covariance), the latent variables were called i-vectors. The i-vector features, extracted using a EM framework with a ML objective, were used for further processing. For example, the speaker verification systems use a probabilistic linear discriminant analysis (PLDA) [15] to model channel variability [16]. The language (accent) recognition systems with i-vectors used a cosine based scoring [17] or a support vector machine (SVM) model for language classification [18]. A modification of prior density was attempted in [19], however the changes to standard normal prior did not show consistent improvements. The baseline system for the LRE 2017 evaluations [20] used the i-vector features with length normalization [21] and linear discriminant analysis [17] followed by a SVM classifier.

Recently, speech recognition systems have also incorporated i-vector features for speaker adaptation [22]. The replacement of GMM-UBM with a deep neural network (DNN) speech recognition front-end has shown improvements in speaker/language recognition tasks [9, 8]. While normalization methods like length normalization have been proposed in the post processing of the i-vectors [21], all the efforts outlined above use the unsupervised ML framework for the training the i-vector models. A previous attempt was made to utilize supervision in i-vector learning by using the label information included with the supervector [23]. However, the approach merely appends the label information (in the form of one-hot encoding) along with the adapted supervector and does not explicitly model the label distribution in a statistical model.

Recently, we have proposed a fully supervised version of the i-vector model [24] where each label class is associated with a Gaussian prior with a class specific mean parameter. The joint prior (marginalized over the sample space of classes) on the latent variable then becomes a GMM. In this paper, we

expand this approach with a detailed account of the EM algorithm for this choice of prior. Specifically, we show that the GMM forms a conjugate prior for this framework (given the statistics, the posterior distribution of the latent vectors is also a GMM). This choice of prior is motivated by the use of a Gaussian back-end [25], where the conventional i-vectors for each language are modeled with a single Gaussian distribution. In the proposed model, the posterior distribution of the i-vectors is a GMM. Thus, the maximum aposteriori (MAP) estimates are not useful for the multi-modal GMM posterior distribution. We resort to the minimum mean square error (MMSE) estimate of the latent variables which we refer as the supervised i-vector (s-vector) for a given test recording. Furthermore, the use of class dependent prior also allows us to weigh the importance of the prior with a factor (the belief on the prior can be varied based on the duration of the recording).

With detailed data analysis and visualization, we show that the s-vector features yield representations that succinctly capture the language (accent) label information. We also show that the conventional i-vectors are a special case of the more generic s-vectors proposed in this work, where all the label information is assumed to belong to one class.

The proposed s-vectors are used for a language recognition task in the NIST Language Recognition Evaluation (LRE) 2017 dataset. We use the same setup as in the baseline LRE system [20] except for the replacement of the unsupervised i-vectors with the s-vectors. The subsequent steps including the SVM training for language classification are also performed. In our experiments comparing the supervised and unsupervised i-vector features, the proposed approach provides significant improvements in LRE task (relative improvements of up to 24% in terms of the primary cost metric). We also show that the proposed approach yields consistent gains for RATS language recognition experiments [26] on short duration conditions.

We also perform accent recognition experiments on the Mozilla Common-Voices dataset [27]. In these experiments, we train and test with very short duration utterances of accented English speech and the task is to recognize the accent of the test utterance. We use s-vector/i-vector based representations with Gaussian backend (GB) classifiers for these experiments.

The rest of the paper is organized as follows. In Sec. 2, we provide the mathematical derivation of the proposed supervised EM framework for s-vector model parameter estimation. We emphasize primarily the difference in the formulation compared to the traditional i-vectors. The minimum divergence re-estimation step is also discussed here. In Sec. 3, we derive the

expressions for s-vector extraction for a given test utterance. The language recognition experiments are reported in Sec. 4. A detailed analysis of the results with confusion matrices and data visualization is provided in Sec. 5. This is followed by a brief summary in Sec. 6.

2. The s-vector model

The i-vector model that is popularly used in speaker and language recognition is outlined in Appendix A. We follow notations similar to the conventional i-vectors to derive the s-vector features. In the traditional i-vector approach, the Total Variability Model (TVM) (Eq. (A.1)) together with the Gaussian Mixture Model - Universal Background Model (GMM-UBM) constitute a latent variable based generative model for the short-time sequence of features. In the s-vector model, we incorporate the class label information in this generative modeling framework. We denote the feature sequence of recording s by

$$X(s) = \{\boldsymbol{x}_1(s), \dots \boldsymbol{x}_{H(s)}(s)\} \tag{1}$$

where $H(s)$ denotes the length of recording s. The corresponding label is denoted as $l(s)$. Note that the values of $l(s)$ are discrete ($l(s) = 1, ..., L$), where L denotes the total number of classes. The adapted means of the GMM-UBM are modeled as:

$$\boldsymbol{M}(s) = \boldsymbol{M}_0 + T\boldsymbol{y}(s) \tag{2}$$

where $\boldsymbol{M}(s)$, $\boldsymbol{M}_0$ and T are similar to the definitions used in the conventional i-vector model (Eq. A.1). Though this equation is identical to the conventional i-vector model, the key difference is that the latent vector $\boldsymbol{y}(s)$ depends on the class label $l(s)$. The prior on $\boldsymbol{y}(s)$ conditioned on the class l is modeled as a Gaussian with mean $\boldsymbol{m}_l$, and a shared, identity covariance matrix for all classes $l \in \{1, \dots, L\}$, i.e.,

$$p(\boldsymbol{y}(s)|l(s) = l) \sim \mathcal{N}(\boldsymbol{y}(s); \boldsymbol{m}_l, I) \tag{3}$$

For recordings without a known label, the prior distribution is then a Gaussian mixture model,

$$\mathrm{p}(\boldsymbol{y}(s)) = \sum_{l=1}^{L} \mathrm{p}(l)\mathcal{N}(\boldsymbol{y} \mid \boldsymbol{m}_l, I) \tag{4}$$

By parameterizing the class conditioned means $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_L$ along with the other model parameters, we make use of the labels of the train recordings to estimate the model parameters, thereby introducing supervision. If $\boldsymbol{m}_l = \boldsymbol{0}$ is set for all $l \in \{1, \ldots, L\}$, the proposed model reverts back to the standard i-vector model. In the s-vector model, we make the following statistical assumptions:

1. The *apriori* probability of observing label l is uniform, i.e., $p(l) = \frac{1}{L} \quad \forall l \in \{1, \ldots, L\}$
2. Given the latent variable $\boldsymbol{y}(s)$, $X(s)$ is conditionally independent on the class label $l(s)$

$$\mathrm{p}_T(l(s) \mid \boldsymbol{y}(s), X(s)) = \mathrm{p}_T(l(s) \mid \boldsymbol{y}(s)) \tag{5}$$

While the full covariance model for each label class is feasible in the proposed framework, the shared covariance model allows for simplicity in model estimation (and greatly reduces the memory requirements) compared to having each mixture class with its own covariance. As some of the classes have a very small number of recordings, the class conditioned covariance matrices may not be well estimated (for example, the language classes like British-English in LRE2017 have very small number of recordings). The popularly used Gaussian backend model [25] for log likelihood computation also doesn't model the covariance of each class separately. The reason for using an Identity covariance matrix is because a model with a shared covariance matrix W can be converted to an equivalent model with identity covariance matrix by the following transformation:

$$\begin{aligned}
\boldsymbol{M}(s) &= \boldsymbol{M}_0 + T\boldsymbol{y}(s) \\
&= \boldsymbol{M}_0 + TW^{\frac{1}{2}}W^{-\frac{1}{2}}\boldsymbol{y}(s) \\
&= \boldsymbol{M}_0 + T'\boldsymbol{y}'(s)
\end{aligned}$$

where $\boldsymbol{y}'$ follows a GMM distribution with Identity mixture covariances. This is achieved through the minimum divergence re-estimation procedure in each iteration (Appendix B.3). A similar approach has been used in the conventional i-vector framework where a standard Gaussian distribution with identity covariance is used instead of a model with full covariance.

We derive the steps involved in the estimation of s-vector model parameters in the following subsection.

2.1. EM Algorithm for Parameter Estimation

The set of parameters in the s-vector model that need to be estimated are denoted as,

$$\Theta = \{T, \boldsymbol{m}_1, \ldots, \boldsymbol{m}_L\}. \tag{6}$$

2.1.1. E-step

For each recording s, using the parameter estimates $\Theta^{(t)}$ at iteration t, we compute the following quantities

$$\mathbb{E}_{\Theta^{(t)}}\left[\boldsymbol{y}(s) \mid X(s), l(s)\right] = \hat{\boldsymbol{y}}^{(t)}_{l(s)}(s) = \boldsymbol{\mathcal{L}}^{(t)}(s)^{-1}\left\{\boldsymbol{m}_{l(s)} + T^{(t)\intercal}\Sigma^{-1}\boldsymbol{F}_X(s)\right\} \tag{7}$$

$$\mathbb{E}_{\Theta^{(t)}}\left[\boldsymbol{y}(s)\boldsymbol{y}(s)^{\intercal} \mid X(s), l(s)\right] = E^{(t)}_{yy,l(s)}(s) = \boldsymbol{\mathcal{L}}^{(t)}(s)^{-1} + \hat{\boldsymbol{y}}^{(t)}_{l(s)}(s)\hat{\boldsymbol{y}}^{(t)}_{l(s)}(s)^{\intercal} \tag{8}$$

where $\boldsymbol{F}_X(s)$ and $N(s)$ are the Baum-Welch statistics in matrix form (defined in Appendix A) and $\boldsymbol{\mathcal{L}}^{(t)}(s) = I + T^{(t)\intercal}\Sigma^{-1}N(s)T^{(t)}$

2.1.2. Maximization Step

The s-vector model parameters are re-estimated using the following update rules. The update equation for the T matrix is

$$T_c^{(t+1)} = \left(\sum_{s=1}^{S} N_c(s)E^{(t)}_{yy,l(s)}(s)\right)^{-1}\left(\sum_{s=1}^{S}\boldsymbol{F}_{X,c}(s)\hat{\boldsymbol{y}}^{(t)}_{l(s)}(s)^{\intercal}\right) \tag{9}$$

where T_c is the sub-matrix of T corresponding to c^{th} mixture of GMM-UBM. The update equations for the class conditioned means are given by

$$\boldsymbol{m}_l^{(t+1)} = \frac{1}{S_l}\sum_{\substack{s=1\\ l(s)=l}}^{S}\hat{\boldsymbol{y}}^{(t)}_{l(s)}(s) \tag{10}$$

where S_l is the number of training recordings with class label l. The detailed derivations of the update equations are given in Appendix B.

2.2. Minimum divergence Re-estimation

The idea of minimum divergence estimation [13] is to model the T matrix in such a way as to force the empirical distribution to conform to the GMM prior as assumed. Specifically, it is required that the class conditioned covariances are shared among all classes and equal to the identity matrix.

We require the within class covariance matrix to be identity. The minimum divergence update equation is given by $T^{(t)} \longleftarrow T^{(t)}L$, where $LL^{\intercal}$ is the Cholesky decomposition of the matrix K_{yy}, given below as,

$$K_{yy}^{(t)} = \frac{1}{S}\sum_{s=1}^{S}\left(\hat{\boldsymbol{y}}_{l(s)}^{(t)}(s)\hat{\boldsymbol{y}}_{l(s)}^{(t)}(s)^{\intercal} - \hat{\boldsymbol{y}}_{l(s)}^{(t)}(s)\boldsymbol{m}_{l(s)}^{(t)\ \intercal} - \boldsymbol{m}_{l(s)}^{(t)}\hat{\boldsymbol{y}}_{l(s)}^{(t)}(s)^{\intercal} + \boldsymbol{m}_{l(s)}^{(t)}\boldsymbol{m}_{l(s)}^{(t)\ \intercal}\right) \tag{11}$$

More details on the above expression are provided in Appendix B.3.

3. S-vector extraction

The conventional i-vectors are simply the maximum *aposteriori* (MAP) estimates of $\boldsymbol{y}(s)$ given the Baum Welch statistics $N(s)$ and $\boldsymbol{F}_X(s)$. In the proposed model, for an unlabeled test recording, the posterior distribution of $\boldsymbol{y}(s)$ turns out to be a GMM (similar to the prior). This model belongs to the broad class of Bayesian models with conjugate prior (The conventional i-vector model is also a conjugate prior Bayesian model, where the prior and posterior are Gaussian distributed). The posterior distribution of $\boldsymbol{y}(s)$ is given by

$$\mathrm{p}(\boldsymbol{y}(s) \mid X(s)) = \sum_{l=1}^{L}\mathrm{p}(l \mid X(s))\,\mathrm{p}(\boldsymbol{y}(s) \mid X(s), l) \tag{12}$$

Given the multi-modal nature of the posterior distribution, there are more than one local maxima (one maximum per label class) to consider if we perform a MAP estimation, and choosing just one of them is tedious when the label information is unknown. One way of deriving i-vector like representations from this model is to splice all the label conditioned MAP estimates to form the recording level representation that can be further dimensionality reduced using principal component analysis. Another option is to perform an average of the individual label conditioned MAP estimates. However, both of these options have statistical limitations as we had observed previously [24].

In this work, we resort to the minimum mean-square-error (MMSE) estimate of $\boldsymbol{y}(s)$ which is referred as the s-vector representation (similar to the MAP estimate in the conventional model which is widely referred to as the

i-vector representation). The MMSE estimate (s-vector) is computed using the following expressions.

$$\hat{\boldsymbol{y}}_{MMSE}(s) = \sum_{l=1}^{L} \mathrm{p}(l \mid X(s))\hat{\boldsymbol{y}}_l(s) \tag{13}$$

We refer to $\hat{\boldsymbol{y}}_l(s)$ as a class conditioned s-vector (conditioned on class l) and it is given by

$$\hat{\boldsymbol{y}}_l(s) = \left(I + T^{\intercal}\Sigma^{-1}N(s)T\right)^{-1}\left(\boldsymbol{m}_l + T^{\intercal}\Sigma^{-1}\boldsymbol{F}_X(s)\right) \tag{14}$$

and the posteriors $\mathrm{p}(l \mid X(s))$ are calculated from the class conditioned likelihoods $\mathrm{p}_\Theta(X(s) \mid l)$. The detailed derivations of above equations are given in Appendix B and Appendix C.

As shown in the experiments, the MMSE based s-vectors perform the best when the posteriors, $p(l \mid X(s))$, are well estimated. Setting $\mathrm{p}(l \mid X(s)) = 1$ for the true class l and 0 for the other classes (true label based one-hot encoding) gives us oracle s-vectors (cheat experiment). Using the oracle s-vectors, we are able to show the performance upper-bounds of the proposed s-vector model. On the other hand, the worst case situation would be when the class posterior distribution is assumed to be uniform ($p(l|X(s)) = \frac{1}{L}$), which are referred to as average s-vectors. The results generated using these average s-vectors are also shown (Sec. 4) in order to illustrate the performance lower bounds in the MMSE estimation.

3.1. Re-weighting the priors

Although the proposed model incorporates label information, the algorithm only tries to maximize the joint likelihood. This does not ensure that the model is discriminative. In an attempt to make the proposed model more discriminative, the class conditioned prior covariance (Eq (3)) can be scaled by a factor $\frac{1}{\lambda}$.

Using the modified prior distribution, it can be shown that the E-step will be modified as follows:

$$\mathcal{L}^{(t)}(s) = \lambda I + T^{(t)\intercal}\Sigma^{-1}N(s)T^{(t)} \tag{15}$$

$$\hat{\boldsymbol{y}}^{(t)}_{l(s)}(s) = \mathcal{L}^{(t)}(s)^{-1}\left(\lambda\boldsymbol{m}^{(t)}_{l(s)} + T^{(t)\intercal}\Sigma^{-1}\boldsymbol{F}_X(s)\right) \tag{16}$$

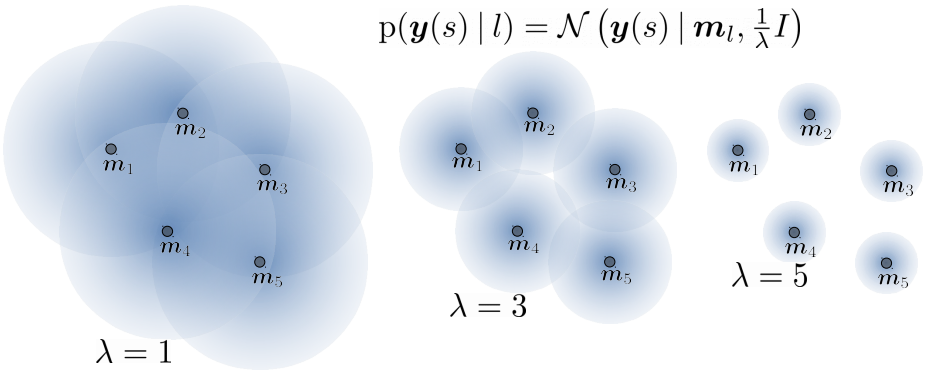

Figure 1: Illustration of the variation of prior distribution with varying λ. In this example, the means of Gaussian mixtures $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_5$ represent five language classes.

With a prior re-weighting factor of λ, the class conditioned s-vector of Eq.(14) will be modified as

$$\hat{\boldsymbol{y}}_l(s) = \left(\lambda I + T^{\intercal}\Sigma^{-1}N(s)T\right)^{-1}\left(\lambda \boldsymbol{m}_l + T^{\intercal}\Sigma^{-1}\boldsymbol{F}_X(s)\right) \tag{17}$$

Figure 1 highlights the effect of λ on the prior density. When the value of λ is increased, the GMM means are unchanged and the mixture component covariance around the means is reduced by a factor of $\frac{1}{\lambda}$. This forces the latent variables to be more concise around the language means thereby enhancing the discriminative ability of the model. However, when a high value of λ is chosen (and in noisy conditions), the embedding could be more concise around the wrong language class mean. Thus, we find that for shorter durations (and in noisy conditions) a lower value of λ is preferred, while for longer durations in cleaner conditions (like in LRE2017), a higher value of λ improves the LID performance. By using the hyper-parameter λ, we can control the degree of confidence on the prior distribution.

4. Experiments and Results

We demonstrate the advantages of the s-vector model by applying it to language and accent recognition problems using the NIST Language Recognition Evaluation 2017 (LRE 2017) dataset [20] and the Mozilla CommonVoice dataset [27] respectively.

4.1. Performance Metrics

We use three different metrics for evaluating the language and accent recognition tasks. The first two metrics, namely the LRE 2017 primary detection cost ($C_{primary}$) and the equal error rates (EER) report the performance on a language detection setting. For this purpose, likelihood ratios are computed for each language versus the rest, and a threshold of β is applied to obtain the false alarm (P_{FA}) and miss (P_{Miss}) probabilities. The detection cost at threshold β is defined as [20]

$$C_{avg}(\beta) = \frac{1}{L}\left\{\sum_{l_T} P_{Miss}(l_T) + \frac{1}{L-1}\left[\beta \times \sum_{l_T}\sum_{l_N \neq l_T} P_{FA}(l_T, l_N)\right]\right\} \quad (18)$$

The detection cost ($C_{Primary}$) used as the primary metric in LRE 2017 is given by

$$C_{Primary} = \frac{C_{avg}(\beta_1 = 1) + C_{avg}(\beta_2 = 9)}{2} \quad (19)$$

The equal error rate (EER) is the P_{FA} (or P_{Miss}) computed at the threshold where P_{FA} and P_{Miss} become equal.

The third metric is the classification accuracy, which reports the performance in a language identification setting (closed set language classification).

4.2. The NIST LRE 2017 task

The NIST Language Recognition Evaluation (LRE) 2017 [20] datasets were used in these experiments. The training dataset consisted of 16205 files from fourteen closely related languages and dialects grouped into five clusters. The details of the dataset are given in Table 1.

The total duration of the train files is about 2069 hours. The development dataset has 3661 files and evaluation dataset has 25451 files. Both dev and eval datasets contain files of duration 3, 10 and 30 seconds from the MLS14 corpus. We trained all the systems using only the LRE 2017 train dataset (LDC2017E22) and we report the performances on the development and evaluation datasets (LDC2017E23).

4.3. The Mozilla Common Voice Dataset

The Mozilla Common Voice is a corpus of speech data read by users [27] based upon text from a number of public domain sources like user submitted blog posts, old books, movies, and other public speech corpora. The dataset contains several different accents of English with total of approximately 64k speech files (sentences) along with the accent labels (Table 2).

Table 1: LRE 2017 training set : Target languages, language clusters, total number of files per language and total duration.

Cluster	Target Languages	#files	Total Duration (hours)
Arabic	Egyptian Arabic (ara-arz)	440	190.9
	Iraqi Arabic (ara-acm)	1406	130.8
	Levantine Arabic (ara-apc)	3509	440.7
	Maghrebi Arabic (ara-ary)	919	81.8
Chinese	Mandarin (zho-cmn)	3331	379.4
	Min Nan (zho-nan)	95	13.3
English	British English (eng-gbr)	98	4.8
	General American English (eng-usg)	2448	327.7
Slavic	Polish (qsl-pol)	587	59.3
	Russian (qsl-rus)	1221	69.5
Iberian	Caribbean Spanish (spa-car)	688	166.3
	European Spanish (spa-eur)	121	24.7
	Latin American Spanish (spa-lac)	898	175.9
	Brazilian Portuguese (por-brz)	444	4.1

We perform the accent recognition task similar to the language recognition pipeline in the NIST LRE 2017 setup. The accents with less than 1 hour were discarded, and 9 accents of English were used in training/test. For this purpose, approximately 100 files from each accent were selected randomly for the development set, and approximately 200 files from each accent were selected randomly for the evaluation set. The remaining files were used for training the accent recognition systems.

4.4. RATS LID Task

The DARPA Robust Automatic Transcription of Speech (RATS) [26] program targets the development of speech systems operating on highly distorted speech recorded over "degraded" radio channels. The data used here consists of recordings obtained from re-transmitting a clean signal over eight different radio channel types, where each channel introduces a unique degradation

Table 2: The Mozilla CommonVoice dataset: Accents chosen, total number of files used for training, development and testing and total duration.

Accent	#files			Total Duration (mins)		
	Train	Dev	Test	Train	Dev	Test
African	926	98	196	61	7.61	14.8
Australian	4182	104	196	239	6.83	12.6
Canada	3778	104	202	229	8.02	14.5
England	15266	96	202	868	6.48	14.2
Indian	4366	103	201	259	7.32	13.8
Ireland	679	102	200	38	6.65	19.3
NewZealand	886	101	204	50	6.68	13.3
Scotland	1323	95	201	75	6.06	13.0
USA	31966	103	198	1845	6.96	13.2

mode specific to the device and modulation characteristics [26]. For the language identification (LID) task, the performance is degraded due to the short segment duration of the speech recordings in addition to the significant amount of channel noise [28]. The training data for the RATS experiments consist of 20000 recordings (about 1600 hours of audio) from five target languages (Arabic, Pashto, Dari, Farsi and Urdu) as well as from several other non-target languages. The development and the evaluation data consists of 5663 and 14757 recordings respectively from the noisy channels. We evaluate the models on 3 sec, 10 sec and 30 sec chunks from the full length evaluation files. All the processing steps involved in i-vector/s-vector extraction for the NIST LRE2017 dataset are used similarly for the RATS dataset.

4.5. Experiments conducted

Figure 2 depicts the pipeline of the bottleneck feature i-vector based language/accent recognition system. In our experiments involving s-vectors, all the components in this pipeline remain the same except for the T matrix training and s-vector extraction steps (described in Sec. 2).

The bottleneck features of 80 dimensions [20] are extracted from a deep neural network (DNN) which was trained on switchboard and Fisher corpora for an automatic speech recognition (ASR) task. Speech activity detection

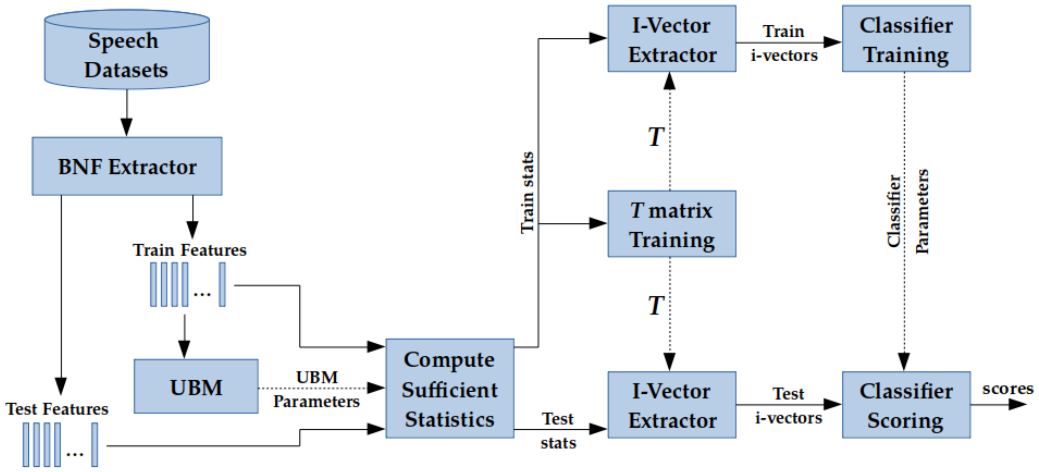

Figure 2: I-vector based language/accent recognition pipeline

(SAD) is applied to retain only the voiced frames. We use the implementation of Sohn's statistical model based VAD from the Voicebox toolkit [29]. A GMM-UBM of 2048 mixtures is trained using the bottleneck features from LRE 2017 train set. For accent recognition experiments involving the CommonVoice datasets, this GMM is trained using the CommonVoice train data. For the total variability training with random initialization, the T matrix is estimated using the EM algorithm for 6 iterations. We set the dimension of the i-vector/s-vector to 500.

Following the i-vector/s-vector extraction, the representations are centered, within class covariance normalized [17], and dimensionality reduced with LDA to $L-1$ dimensions ($L = 14$ for LRE and 9 for the accent recognition task). We then use two different back-end models for obtaining the language log-likelihood scores, namely the Gaussian backend (GB) [25] and Support Vector Machines (SVM) [18] with radial basis function (RBF) kernel. It was observed that SVM outperformed GB for the LRE task and the GB outperformed SVM for the accent recognition task using the baseline i-vector features. Hence, for purpose of conciseness, we report only the performance of the SVM based backend for LRE task and only the GB based back-end for the accent recognition task. For the RATS LID experiments, we use the i-vector/s-vector embeddings with a support vector machine back-end using a polynomial kernel [28].

Table 3: Results on LRE Development Dataset using SVM back-end for scoring

	Dev Performances : $100C_{primary}$ [EER (%)] {Accuracy(%)}		
Model config.	3 sec	10 sec	30 sec
Unsupervised i-vector [20]	52.7 [16.6] {51.8}	27.1 [7.5] {74.0}	13.1 [3.6] {87.8}
Sup. i-vector [23]	47.2 [15.1] {61.1}	20.2 [5.7] {80.9}	14.8 [4.1] {85.1}
Simplified Sup. i-vector [23]	58.2 [19.6] {47.8}	27.3 [7.8] {73.6}	13.4 [3.7] {87.7}
Average s-vector	47.8 [14.3] {56.3}	22.4 [6.2] {78.3}	12.2 [**3.3**] {88.0}
PCA s-vector [24]	57.6 [18.2] {50.4}	27.0 [7.4] {74.6}	12.1 [3.4] {88.2}
LSTM [30]	53.7 [15.39] {52.7}	33.8 [9.7] {69.2}	32.0 [8.81] {70.85}
HGRU [31]	53.1 [15.09] {57.5}	27.6 [6.6] {76.9}	25.5 [6.1] {78.6}
MMSE s-vector	**44.4** [**14.7**] {**63.5**}	**19.5** [**5.9**] {**83.7**}	**11.7** [3.4] {**89.4**}
Oracle s-vector (cheat)	13.0 [3.9] {88.3}	5.6 [1.7] {94.4}	5.0 [1.1] {94.9}

4.6. Results on LRE experiments

We use $C_{primary}$ as the primary metric to compare the performance of various language and accent recognition systems. In this study, we vary the value of λ, and see how $C_{primary}$ varies with λ for the development dataset, for each duration (3 sec, 10 sec and 30 sec) separately. The value of λ that gives the best performance on the development set is then used to compare our s-vector systems with the baseline on the evaluation set. We also report the EER and accuracy of the systems evaluated.

Figure 3 shows the variation of $C_{primary}$ with λ for MMSE s-vector systems on the LRE 2017 development dataset. As seen in the figure, the optimal choice of λ was 3, 5 and 7 for 3 sec, 10 sec and 30 sec conditions respectively. These configurations are used on the evaluation dataset.

In Tables 3 and 4, we report the $C_{primary}$, equal error rates (EER) and classification accuracies for the baseline systems, namely the unsupervised i-vectors systems, simplified supervised i-vector systems [23], and our s-vector systems. As noted previously, the oracle s-vectors (cheat) and the average s-vectors represent performance bounds.

On the LRE evaluation dataset the proposed s-vector improves over the baseline relatively by [19%, 20%, 8%] in terms of $C_{primary}$, [16%, 24%, 4%] in terms of EER, and [16%, 18%, 9%] in terms of accuracy for [3 sec, 10 sec, 30 sec] durations.

4.7. Results on Accent Recognition Task

The following figure shows the variation of $C_{primary}$ with λ for MMSE s-vectors on the development dataset using Gaussian Backend classifiers. In

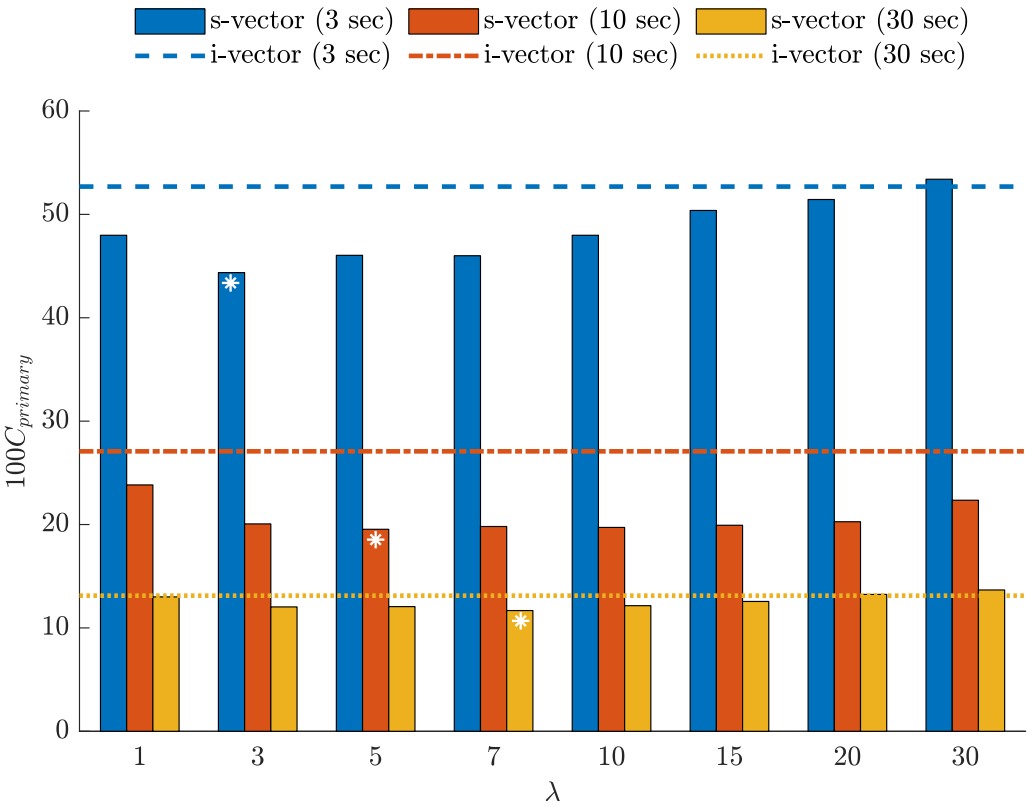

Figure 3: Variation of $C_{primary}$ with λ for MMSE s-vectors on development data for each of the durations [3 sec, 10 sec, and 30 sec]. The dotted line denotes the unsupervised i-vector baseline.

this case, the MMSE s-vector ($\lambda = 7$) gave the best performance on development data. This configuration is used on the evaluation set and the results are reported in Table 5. On the evaluation dataset, the proposed s-vectors provide only moderate improvements (over the baseline), with average relative improvements of 2% in terms of $C_{primary}$, 2.4% in terms of EER, and 4% in terms of accuracy. As the recordings are very short in duration and the accent classes are highly overlapping, the posterior distribution is not well estimated. Hence, the average s-vector also performs as good as the MMSE s-vector for this task.

Table 4: Results on LRE Evaluation Dataset using SVM back-end for scoring and other neural network based approaches.

	Eval Performances : $100C_{primary}$ [EER (%)] {Accuracy(%)}		
Model config.	3 sec	10 sec	30 sec
Unsupervised i-vector [20]	53.6 [16.1] {53.8}	29.9 [8.6] {72.4}	16.7 [3.9] {83.0}
Sup. i-vector [23]	46.1 [14.7] {59.6}	25.6 [7.3] {76.4}	19.9 [5.1] {80.9}
Simplified Sup. i-vector [23]	57.2 [19.1] {49.8}	29.8 [8.4] {71.4}	16.7 [4.1] {82.8}
LSTM [30]	55.2 [15.4] {54.7}	35.4 [8.7] {72.1}	28.1 [7.3] {76.1}
HGRU [31]	55.4 [15.3] {55.1}	32.3 [7.5] {74.1}	23.3 [4.9] {83.0}
Average s-vector	49.7 [13.9] {58.5]	27.0 [7.0] {75.3}	15.7 [**3.7**] {84.0}
PCA s-vector [24]	58.2 [7.7] {54.1}	30.5 [8.2] {73.7}	15.8 [3.8] {83.9}
MMSE s-vector	**43.7** [**13.5**] {**61.2**}	**23.7** [**6.5**] {**77.4**}	**15.4** [3.8] {**84.5**}
Oracle s-vector (cheat)	12.6 [3.6] {86.9}	6.5 [1.6] {92.9}	5.7 [1.4] {93.6}

4.8. Results on RATS Language Recognition Task

The RATS LID results are reported in Table 4.7. The RATS dataset involves language recognition on 5 target languages along with several other imposter classes. The proposed s-vector approaches show consistent improvements over the baseline system on short duration conditions (3 sec. and 10 sec.) over the baseline i-vector approach. For example, on the RATS evaluation set for the 3 sec. and 10 sec. condition, the proposed s-vector approach improves the baseline system by about 11 % relative in terms of $C_{primary}$ metric. These results are also consistent with the NIST LRE2017 results.

4.9. Computational Complexity

With R being the dimension of the i-vectors/s-vectors and if L labels are included in the model, the complexity of both i-vector and s-vector extraction is of order $O(R^3)$. The s-vector extraction involves an additional step of computing the posterior probability $\mathrm{p}(l \mid X)$ and the language specific posterior mean for each langauge l. Both of these steps are of the order $O(RL)$. In order to analyze the computation time, 50 files of 30 sec. duration from the NIST LRE task were selected and their i-vectors and s-vectors were extracted sequentially in a single threaded mode on an Intel CPU with 256 GB of RAM. The feature extraction and zeroth/first order statistic computation was performed as a pre-processing step before the i-vector/s-vector estimation. The total computation time for 50 recordings was about 24 sec. and 28 sec. respectively for the i-vector and s-vector estimation procedure.

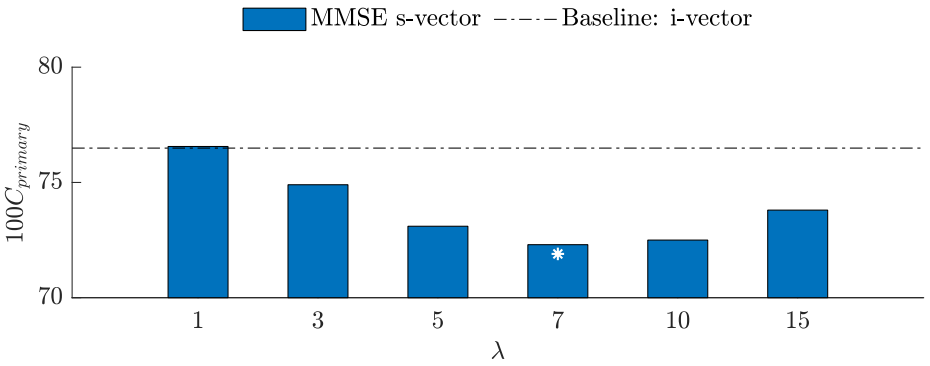

Figure 4: Variation of $C_{primary}$ with λ for the accent recognition task on the Mozilla Commons Development Dataset.

Table 5: Results on accent recognition experiments with Mozilla CommonVoice Datasets using Gaussian backend for scoring

Model config.	Eval Performances: $100C_{primary}$ [EER (%)] {Accuracy (%)}
Unsupervised i-vector [20]	77.2 [21.2] {52.5}
Average s-vector	**75.9** [20.7] {**53.3**}
MMSE s-vector	76.0 [**20.7**] {53.2}
Oracle s-vector (cheat)	61.5 [15.3] {62.2}

Thus, the s-vector estimation involves 15 % more computation time than the i-vector estimation. However, we find that the effect of this increased computation impacts the overall processing pipeline involving voice activity detection, feature extraction, embedding estimation and SVM scoring by only about 2 % relative.

5. Discussion

5.1. Result Analysis on the LRE task

For the LRE task, the s-vectors show significant improvement over the i-vector baseline on the dev data for most choices of λ (Figure 3). As seen in Figure 1, increasing the value of λ improves the belief on the class distribution chosen by the posterior ($p(l|X(s))$) (the Gaussian clusters are more concise around the mean in Figure 1). For short duration conditions like 3 sec, the posterior distribution $p(l|X(s))$ is not well estimated and has errors. Hence,

Table 6: Results on RATS Language identification systems

	Performances : $100C_{primary}$ [EER (%)] {Accuracy(%)}		
Model config.	3 sec	10 sec	30 sec
	Dev.		
Unsup. i-vector [20]	94.0 [26.3] {66.1}	63.8 [17.6] {77.1}	**40.5** [10.9] {86.9}
MMSE s-vector	**86.7** [25.2] {70.2}	**59.5** [16.6] {80.2}	41.3 [11.3] {87.5}
	Eval		
Unsup. i-vector [20]	89.0 [25.1] {64.8}	63.0 [17.9] {76.2}	40.3 [11.6] {85.7}
MMSE s-vector	**79.5** [22.5] {65.8}	**56.3** [16.3] {77.5}	**39.5** [11.4] {85.4}

increasing the value of λ makes the distribution concise on poorly estimated posteriors which degrades the performance. On the other hand, for the longer duration condition of 30 sec, the posterior $p(l|X(s))$ is well estimated and having a concise distribution (by increasing the value of λ) improves the performance. Thus, there is a trade-off in the choice of λ that can provide the optimal performance based on the duration of the utterance. For shorter duration utterances like 3 sec, the best performance is achieved at $\lambda = 3$, for moderately long duration utterances (10 sec), the best performance is achieved by using $\lambda = 5$, and for long recordings of 30 sec, the best choice is $\lambda = 7$. Also, the relative improvements for the proposed s-vector approach over the conventional i-vector system are more significant on the 3 sec and 10 sec conditions which are more challenging.

We also compare the performance of the proposed s-vectors with supervised i-vectors and simplified supervised i-vectors introduced in [23] (Table 3, 4). This previous approach of using language labels fails to statistically model the label distribution as the label information (in the form of one-hot encoded vectors) is appended to the adapted means of the GMM (without any change in the i-vector modeling framework). In the proposed work, the labels are handled as discrete symbols and the label information impacts the choice of the prior distribution in the EM framework. The results indicate that the proposed approach of using labels is superior to the previous work [23] for all durations.

The results for neural network based approaches for language recognition in LRE2017 development and evaluation data [30, 32, 31] are also reported in Table 4). The long short term memory (LSTM) recurrent neural network (RNN) based LID system [30, 32] uses an end-to-end LSTM model for lan-

guage recognition. An end-to-end hierarchical model for language recognition [31] using gated recurrent units (GRU) improved the LSTM based model for longer duration speech recordings. Both these models, use the same training and test data compared to the proposed s-vector model. As seen in Table 3 and Table 4, the s-vector model improves over the baseline neural network models in all test duration conditions .

The proposed model involves a Bayesian estimation of embedding vectors $\boldsymbol{y}(s)$ using data statistics $X(s)$ and a prior distribution $p(\boldsymbol{y}(s)$ which is assumed to be a Gaussian mixture model (GMM). The baseline approach (based on standard i-vector framework) also involves a Bayesian estimation using a standard Gaussian prior. As the duration of given utterance increases (from 3 sec. to 30 sec.), the data statistics $X(s)$ are well estimated and the prior information has a vanishing effect in the latent vector estimation. Given that the difference between the two models (baseline and the proposed approach) is in the choice of prior density, we hypothesize that the proposed approach has more significant improvements on the short duration data (where the prior information has a dominant effect). We also re-weight the prior to increase the significance of the prior density in the embedding vector estimation using the hyper-parameter λ.

5.2. Data Visualization

In order to analyze the improvements obtained using the proposed approach, we use a data visualization approach using the t-distributed stochastic neighborhood embedding (tSNE) [33]. The tSNE is an unsupervised dimensionality reduction method which preserves local neighborhood of the data space in the lower dimensional subspace. We perform tSNE dimensionality reduction to two dimensions on the unsupervised i-vectors and the proposed s-vectors (on the LRE development set for 3 sec recordings). The two dimensional scatter plot is separately shown for each of the five language clusters (Arabic, Chinese, English, Slavic and Iberian). This is because most of the confusions in language classification happen within the broad language cluster. The tSNE plots are shown in Figure 5. As seen here, for most of the language clusters, the s-vectors have a reduced within class variance in the cluster distribution. For English and Chinese languages, the between class separability is also improved for the proposed s-vectors. The tSNE plots illustrate that the s-vectors provide representations that are better suited for language recognition compared to the unsupervised i-vectors.

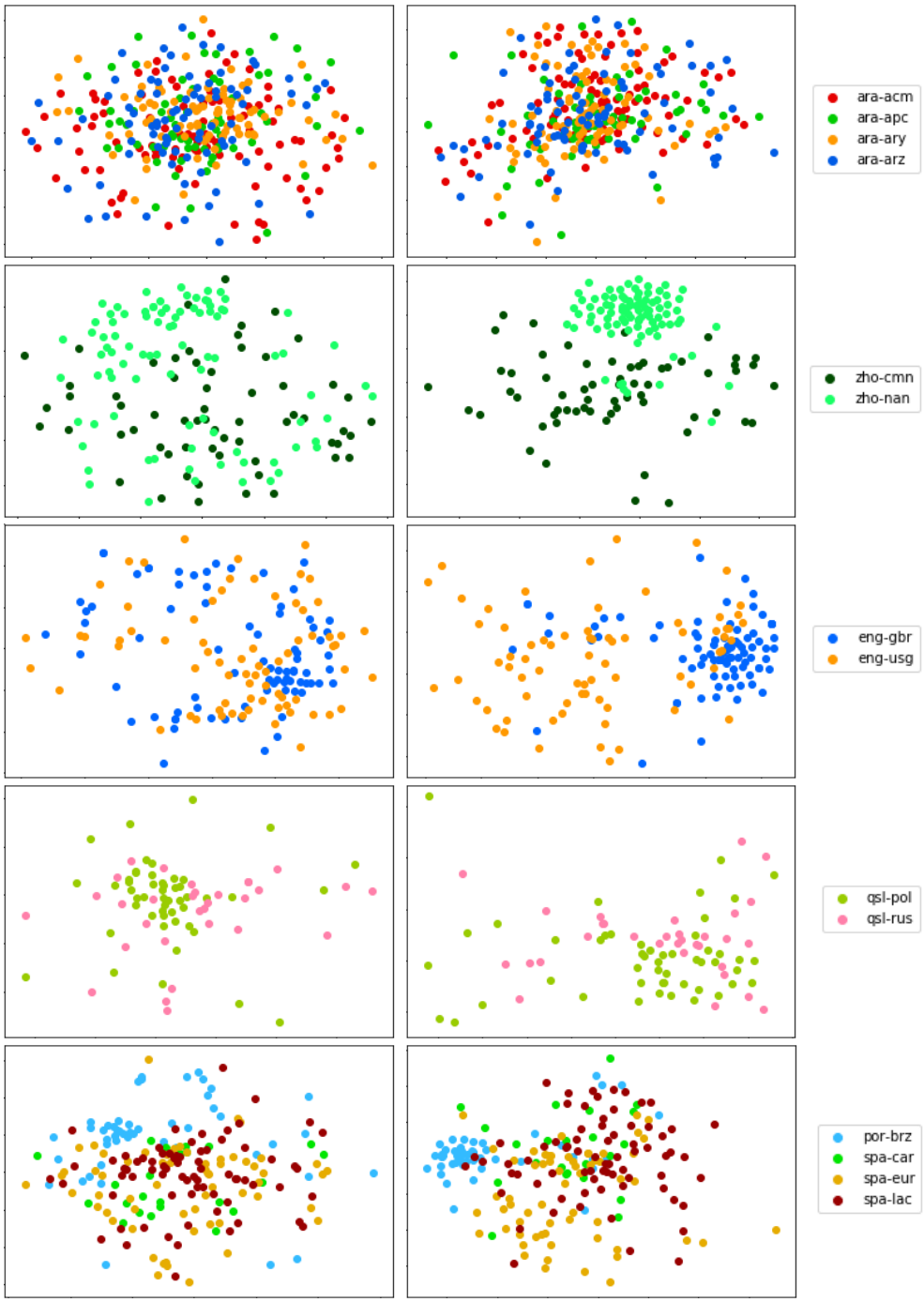

Figure 5: t-SNE scatter plots of unsupervised i-vectors (left) and MMSE s-vectors with $\lambda = 3$ (right) for the LRE 2017 development dataset with 3 sec recordings. The language clusters belong to Arabic, Chinese, English, Slavic and Iberian (from top to bottom).

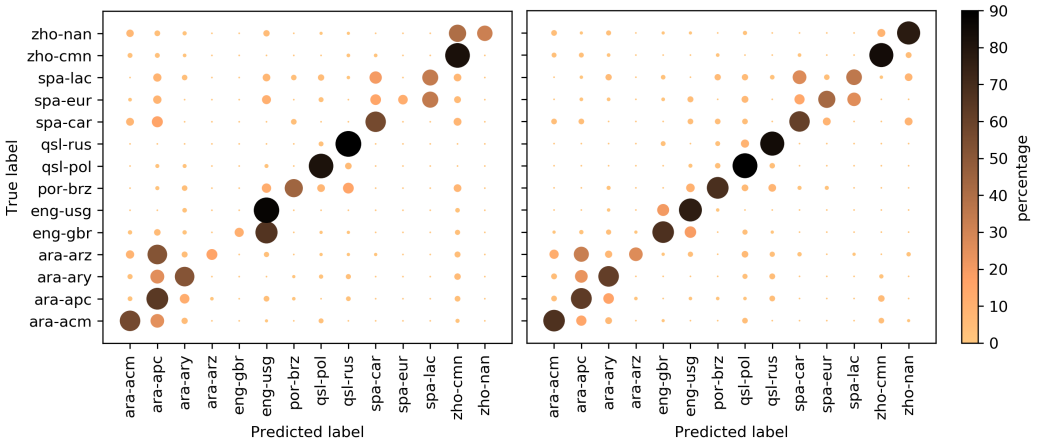

Figure 6: Row-normalized Confusion Matrices of i-vector system (left) and s-vector system with $\lambda = 3$ (right) on LRE 2017 development dataset for the 3 sec condition.

5.3. Confusion Matrix Analysis

The row-normalized confusion matrix plots for 3 sec recordings in the LRE development set is shown in Figure 6. The ideal confusion matrix plot is a identity matrix where all the non-diagonal entries are zeros and the matrix is diagonally dominant. The confusion matrix plot of the baseline system (left side plot) indicates that for many languages like eng-gbr, ara-arz and spa-eur the diagonal entries are not the highest in the row (indicating that majority of the examples of the particular language class are confused as another class within the same broad language cluster). While this issue persists for the ara-arz class in the proposed s-vector model (right side plot), all the other language classes have the desired diagonal dominance. In particular, the two language clusters that showed a good class separation in the tSNE plots (Figure 5) for the s-vector model (eng-usg versus eng-gbr and zho-cmn versus zho-nan) also showed significantly reduced confusions(Figure 6). Part of the degradation of the baseline system on these dialects of English and Chinese language cluster may be attributed to the highly imbalanced training data for these languages (Table 1). Thus, the confusion plots highlight that the proposed model not only improves the overall performance, but is able to improve the class-specific performances on challenging cases where the training data is somewhat limited.

5.4. Relationship to Prior Work

In this paper, we have proposed the s-vector representation. This is based on MMSE estimation of latent representations that have a GMM prior density in the factor analysis model. In the original i-vector model [13], a standard Gaussian density is used. The motivation of standard Gaussian density in the factor model is two fold - i) the Gaussian prior density results in a conjugate prior where the posterior density is also Gaussian distributed, ii) the MAP estimate is simply the posterior mean and that allows efficient estimation of latent representation for speaker/language recognition. However, the standard i-vector model is unsupervised and does not use the language labels of the training dataset even when they are available for the language recognition task. The proposed approach overcomes this limitation by using a GMM prior density for the latent representation. The mixture components correspond to the target language classes. Comparing with the standard i-vector model, the proposed approach also yields a conjugate posterior density. However, the simple MAP estimation is no longer feasible and more involved minimum-mean square error (MMSE) estimate is required to obtain the latent s-vector representations. We have highlighted mathematical formulation to generate the s-vector representations using MMSE approach in Sec. 2 and Appendix B. Thus, with moderate increase in computational complexity, we show that the proposed approach is able to efficiently incorporate the label information in the training data for the embedding extractor.

The use of GMM prior density has been attempted in the past in [19, 34, 35]. While the modeling strategy in these works are similar to the proposed approach, they make approximations to simplify the posterior estimation that defy the underlying the Bayesian factor analysis model assumptions.

In [19], the authors use the GMM prior density with each mixture component corresponding to one of the language class labels similar to the proposed approach. However, the authors approximate the posterior as shown below,

$$\begin{aligned} p(\boldsymbol{y}(s)|X(s)) &= \sum_{l=1}^{L} p(\boldsymbol{y}(s), l|X(s)) \\ &= \sum_{l=1}^{L} p(l|X(s), \boldsymbol{y}(s)) p(\boldsymbol{y}(s)|l) \end{aligned} \tag{20}$$

The posterior $p(l|X(s), y(s))$ is assumed as $p(l)$ which is the prior probability of the language label [19]. This serves as an approximation at best and

Table 7: Results on NIST LRE 2017 Evaluation Dataset for two conditions where the λ is fixed and when the λ is tied to the trace of the posterior covariance matrix.

	Performances : $100C_{primary}$ [EER (%)] {Accuracy(%)}		
Model config.	3 sec	10 sec	30 sec
MMSE s-vector (λ fixed)	43.7 [13.5] {61.2}	23.7 [6.5] {77.4}	**15.4** [3.8] {84.5}
MMSE s-vector (λ tied)	**43.6** [13.8] {61.5}	24.7 [6.8] {77.2}	15.5 [3.9] {84.5}

violates the Bayesian posterior probability model. The resulting embedding used in [19] ($E(\boldsymbol{y}(s)|X(s))$ with this approximation is not a MMSE estimate of the latent vector. In our work, we derive the exact expression for $\mathrm{p}(l|X(s))$ under the specific case where the prior covariances of each class (mixture component) are shared and equal to $\frac{1}{\lambda}I$ (Eq. C.5). Then, we proceed to use the posterior to find the MMSE estimate $E(\boldsymbol{y}(s)|X(s))$.

The methods developed in [34, 35], use a GMM prior density on the latent representations where the mixture components correspond to phonetic groups. The authors use an external phonetic recognizer (Hungarian phoneme recognition system) to obtain frame-level phoneme posteriors. These frame level posteriors are converted to utterance level posteriors using an accumulation of frame level posterior statistics. In our approach, we do not employ an external model for posterior estimation as they are directly obtained from the s-vector model itself. In addition, the proposed approach has utterance level language labels which is different from the frame level phoneme labels used in [34, 35]. Hence, there is no approximation needed to convert posterior information from frame level to utterance level.

The MMSE approach also addresses a crucial drawback of our previous work [24] in the estimation of the embedding vector (s-vector). The previous work [24] using PCA does not estimate any posterior probability and merely computes the i-vector representation for each language class. These language specific i-vectors are concatenated and a PCA is applied for dimensionality reduction. In this work, a full Bayesian estimation of the embedding vector is proposed where the estimation is approached using a MMSE criterion. The proposed approach is more elegant mathematically while it also simplifies the computation over the PCA method. In addition, as seen in results in Table 3 and Table 4, the proposed s-vector approach improves significantly over PCA based embedding vectors.

Table 8: Performance in terms of equal error rate (EER) % for a closed set speaker recognition experiment on the Librispeech dataset.

Model config.	50 spk.	100 spk.	200 spk.
Unsup. i-vector	0.122	0.156	0.215
MMSE s-vector	0.122	0.158	0.218

5.5. Estimating the Prior Weight Using Posterior Covariance

In the language recognition experiments reported in Table 3 and Table 4, the hyper-parameter λ that controls the weighting of the prior (covariance matrix of the prior density is $\frac{1}{\lambda}I$) is chosen based on the performance in development data. In a subsequent analysis, we attempt the estimation of the hyper-parameter λ as function of the trace of the posterior covariance $\mathcal{L}^{(t)}(s)^{-1}$. We use a second order polynomial (obtained using the development set) to find the value of λ for each utterance. Note that, the prior-weighting changes for each utterance in this case and it is tied to the posterior covariance unlike the fixed choice of λ per duration used in the previous experiments. Table 7 compares the results on NIST LRE2017 evaluation dataset for the two cases 1) with fixed λ that is duration specific, 2) with utterance level choice of λ that is tied to the trace of the posterior covariance matrix. As seen in this Table, the language recognition performance is similar in both cases indicating that prior density covariance parameter λ can be chosen based on the statistics of the data for each utterance. The approach of tying the λ value to the posterior covariance is partly motivated by previous efforts on uncertainty propagation in factor analysis [36].

5.6. Application to Closed Set Speaker Recognition

One of the potential drawbacks of the proposed approach is the reliance on the supervised labels in the embedding extraction. For language recognition tasks, the number of class labels are typically small thereby allowing the modeling of each language class with a GMM component. In tasks such as speaker recognition, where i-vector approaches are dominantly used, the number of class labels (speakers) can be significantly high. In order to test the limits of the proposed approach for cases with large number of classes, we perform a closed set speaker recognition task on the librispeech dataset [37]. Here, we train a background model and the total variability matrix from a

set of background speakers (from the librispeech dataset). The test dataset consists of varied number of speakers (50, 100 and 200) and the multi-class SVM is used as back-end model for speaker recognition. The i-vector/s-vector embeddings are used in these experiments and the performance is measured in terms of equal error rate (EER). Table 8 reports the results for speaker recognition experiment on librispeech dataset. As seen in Table 8, the s-vector system did not improve over the i-vector approach in the closed set speaker recognition task. However, even with 200 speaker classes, the performance of the proposed s-vector model does not degrade compared to the i-vector approach. One potential future research direction for speaker verification would be to use an unsupervised speaker clustering approach to generate the label classes for the s-vector model. This may reduce the number of classes while still preserving the speaker discriminability used in the s-vector model.

6. Summary

The major contributions from this work are summarized as follows,

- We have established the theory for supervised i-vector (s-vector) modeling where the label information is incorporated in the prior density modeling.
- We have proposed the use of a hyper-parameter λ to control the degree of importance assigned to prior distribution in the MMSE estimation of s-vector representations.
- The conventional i-vector model is shown to be a special case of the proposed model where the class conditioned means $\boldsymbol{m}_l$ are forced to $\mathbf{0}$ for all language classes.
- The benefits of the proposed s-vector modeling are illustrated using the NIST LRE 2017 task where the s-vector approach is compared with the unsupervised i-vectors. The proposed approach yields significant improvements in terms of LRE primary cost metric for all duration specific conditions considered. The results on a separate accent recognition task further highlight the advantages of the proposed s-vector framework.

- A detailed analysis using data visualization and confusion matrices shows that the s-vector representations improve the separability of dialect variations with the same broad language cluster where the training data is highly imbalanced.

Acknowledgements

This work was funded partly by grants from the Department of Science and Technology Extra Mural Research Grant (EMR/2016/007934) and Pratiksha Trust Young Scientist Research Grant. The authors would like to acknowledge Bharat Padi, Anand Mohan, Satish Kumar and Vaishnavi Y for their help in setting up the LRE 2017 baseline, Omid Sadjadi for code fragements implementing the NIST LRE baseline system and Ming Li for the code implementing the simplified supervised i-vector system.

7. References

[1] M. A. Zissman, Comparison of four approaches to automatic language identification of telephone speech, IEEE Transactions on Speech and Audio Processing 4 (1996) 31–44.

[2] J. Navratil, Spoken language recognition-a step toward multilinguality in speech processing, IEEE Transactions on Speech and Audio Processing 9 (2001) 678–685.

[3] N. Brümmer, S. Cumani, O. Glembek, M. Karafiát, P. Matejka, J. Pesán, O. Plchot, M. Soufifar, E. de Villiers, J. Cernocký, Description and analysis of the Brno276 system for LRE2011, in: Odyssey Speaker and Language Recognition Workshop (2012), pp. 216–223.

[4] S. Davis, P. Mermelstein, Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, IEEE Transactions on Acoustics, Speech, and Signal Processing 28 (1980) 357–366.

[5] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, J. R. Deller Jr, Approaches to language identification using Gaussian mixture models and shifted delta cepstral features, in: Seventh International Conference on Spoken Language Processing (2002), pp. 89–92.

[6] M. F. BenZeghiba, J.-L. Gauvain, L. Lamel, Phonotactic Language Recognition Using MLP Features, in: INTERSPEECH (2012), pp. 2041–2044.

[7] J. Ma, B. Zhang, S. Matsoukas, S. Mallidi, F. Li, H. Hermansky, Improvements in language identification on the RATS noisy speech corpus, in: INTERSPEECH (2013), pp. 69–73.

[8] F. Richardson, D. Reynolds, N. Dehak, Deep neural network approaches to speaker and language recognition, IEEE Signal Processing Letters 22 (2015) 1671–1675.

[9] Y. Lei, N. Scheffer, L. Ferrer, M. McLaren, A novel scheme for speaker recognition using a phonetically-aware deep neural network, in: International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 1695–1699.

[10] S. Ganapathy, K. Han, S. Thomas, M. Omar, M. V. Segbroeck, S. S. Narayanan, Robust language identification using convolutional neural network features, in: INTERSPEECH, pp. 1846–1850.

[11] D. A. Reynolds, R. C. Rose, Robust text-independent speaker identification using Gaussian mixture speaker models, IEEE Transactions on Speech and Audio Processing 3 (1995) 72–83.

[12] D. A. Reynolds, T. F. Quatieri, R. B. Dunn, Speaker verification using adapted Gaussian mixture models, Digital signal processing 10 (2000) 19–41.

[13] P. Kenny, Joint factor analysis of speaker and session variability: Theory and algorithms, (Report) CRIM 14 (2006) 1–17.

[14] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, P. Ouellet, Front-end factor analysis for speaker verification, IEEE Transactions on Audio, Speech, and Language Processing 19 (2011) 788–798.

[15] S. J. Prince, J. H. Elder, Probabilistic linear discriminant analysis for inferences about identity, in: International Conference on Computer Vision (ICCV) (2007), pp. 1–8.

[16] L. Burget, O. Plchot, S. Cumani, O. Glembek, P. Matějka, N. Brümmer, Discriminatively trained probabilistic linear discriminant analysis for speaker verification, in: International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2011), IEEE, pp. 4832–4835.

[17] N. Dehak, P. A. Torres-Carrasquillo, D. Reynolds, R. Dehak, Language recognition via i-vectors and dimensionality reduction, in: INTERSPEECH (2011), pp. 857–860.

[18] W. M. Campbell, E. Singer, P. A. Torres-Carrasquillo, D. A. Reynolds, Language recognition with support vector machines, in: Odyssey: The Speaker and Language Recognition Workshop (2004), pp. 285–288.

[19] R. Travadi, M. V. Segbroeck, S. S. Narayanan, Modified-prior i-vector estimation for language identification of short duration utterances, in: INTERSPEECH (2014), pp. 3037–3041.

[20] S. O. Sadjadi, et al., The 2017 NIST language recognition evaluation, in: Odyssey Speaker and Language Recognition Workshop (2018), pp. 82–89.

[21] D. Garcia-Romero, C. Y. Espy-Wilson, Analysis of i-vector length normalization in speaker recognition systems, in: INTERSPEECH (2011), pp. 249–252.

[22] G. Saon, H. Soltau, D. Nahamoo, M. Picheny, Speaker adaptation of neural network acoustic models using i-vectors, in: ASRU (2013), pp. 55–59.

[23] M. Li, S. Narayanan, Simplified supervised i-vector modeling with application to robust and efficient language identification and speaker verification, Computer Speech & Language 28 (2014) 940–958.

[24] S. Ramoji, S. Ganapathy, Supervised i-vector modeling-theory and applications, in: INTERSPEECH (2018), pp. 1091–1095.

[25] D. Martinez, O. Plchot, L. Burget, O. Glembek, P. Matějka, Language recognition in ivectors space, in: INTERSPEECH (2011), pp. 861–864.

[26] K. Walker, S. Strassel, The rats radio traffic collection system, in: Odyssey 2012-The Speaker and Language Recognition Workshop.

[27] Mozilla Common Voice Dataset, `https://voice.mozilla.org/en/datasets`, 2018. [Online; accessed 19-July-2018].

[28] K. J. Han, S. Ganapathy, M. Li, M. Omar, S. Narayanan, TRAP Language identification system for RATS phase II evaluation, in: Interspeech, ISCA.

[29] J. Sohn, N. S. Kim, W. Sung, A statistical model-based voice activity detection, IEEE signal processing letters 6 (1999) 1–3.

[30] R. Zazo, A. Lozano-Diez, J. Gonzalez-Rodriguez, Evaluation of an lstm-rnn system in different nist language recognition frameworks, in: Proc. of Odyssey 2016 Speaker and Language Recognition Workshop, ATVS-UAM.

[31] B. Padi, A. Mohan, S. Ganapathy, End-to-end language recognition using attention based hierarchical gated recurrent unit models, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 5966–5970.

[32] R. Zazo, A. Lozano-Diez, J. Gonzalez-Dominguez, D. T. Toledano, J. Gonzalez-Rodriguez, Language identification in short utterances using long short-term memory (lstm) recurrent neural networks, PloS one 11 (2016) e0146917.

[33] L. Maaten, G. Hinton, Visualizing data using t-SNE, Journal of machine learning research 9 (2008) 2579–2605.

[34] J. Ma, V. Sethu, E. Ambikairajah, K. A. Lee, Speaker-phonetic vector estimation for short duration speaker verification, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 5264–5268.

[35] J. Ma, V. Sethu, E. Ambikairajah, K. A. Lee, Generalized variability model for speaker verification, IEEE Signal Processing Letters 25 (2018) 1775–1779.

[36] C. Yu, G. Liu, S. Hahm, J. H. Hansen, Uncertainty propagation in front end factor analysis for noise robust speaker recognition, in: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 4017–4021.

[37] V. Panayotov, G. Chen, D. Povey, S. Khudanpur, Librispeech: an asr corpus based on public domain audio books, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 5206–5210.

Appendix A. An account of the conventional i-vector Model

We provide the mathematical derivation of the conventional i-vector modeling [13]. This is required as the s-vector modeling is built on this model. The notations used here follow those from the work of Kenny and Dehak et. al. [13, 14].

Given a dataset of S recordings, let $X(s) = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{H(s)}]$ denote the sequence of F dimensional front-end feature vectors, where $H(s)$ is the number of frames in recording s. Let $\Lambda = \{\pi_c, \boldsymbol{\mu}_c, \Sigma_c\}_{c=1}^{C}$ denote the parameters of a C component Gaussian Mixture Model - Universal Background Model (GMM-UBM). The UBM mean supervector is denoted by $\boldsymbol{M}_0 = [\boldsymbol{\mu}_1{}^{\intercal}, \ldots, \boldsymbol{\mu}_c{}^{\intercal}]^{\intercal}$. It is assumed that for each recording s, an adapted mean supervector $\boldsymbol{M}(s)$ was used to generate $X(s) = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{H(s)}]$. The i-vector model (also known as the total variability model) is a generative model for $\boldsymbol{M}(s)$ and is given by

$$\boldsymbol{M}(s) = \boldsymbol{M}_0 + T\boldsymbol{y}(s) \tag{A.1}$$

where T is a matrix of dimension $CF \times R$ called the total variability matrix, and $\boldsymbol{y}(s)$ is a latent vector of dimension $R \times 1$. A standard normal distribution is used as the prior density for $\boldsymbol{y}(s)$. The i-vector of recording s is defined as the MAP estimate of $\boldsymbol{y}(s)$ given $X(s)$. The Baum-Welch statistics of recording s for mixture c are given by

$$N_c(s) = \sum_{i=1}^{H(s)} \mathrm{p}_\Lambda(c \mid \boldsymbol{x}_i) \tag{A.2}$$

$$\boldsymbol{F}_{X,c}(s) = \sum_{i=1}^{H(s)} \mathrm{p}_\Lambda(c \mid \boldsymbol{x}_i)(\boldsymbol{x}_i - \boldsymbol{\mu}_c) \tag{A.3}$$

$$S_{XX,c}(s) = \sum_{i=1}^{H(s)} \mathrm{p}_\Lambda(c \mid \boldsymbol{x}_i)(\boldsymbol{x}_i - \boldsymbol{\mu}_c)(\boldsymbol{x}_i - \boldsymbol{\mu}_c)^{\intercal} \tag{A.4}$$

The BW statistics are sufficient statictics for estimation of the model parameters. In matrix form, the zeroth and first order statistics are written as:

$$N(s) = \begin{pmatrix} N_1(s)I & & 0 \\ & \ddots & \\ 0 & & N_C(s)I \end{pmatrix}, \qquad \boldsymbol{F}_X(s) = \begin{pmatrix} \boldsymbol{F}_{X,1}(s) \\ \vdots \\ \boldsymbol{F}_{X,C}(s) \end{pmatrix}$$

where I is an identity matrix of size $F \times F$. It can be shown that the log-likelihood function of $\boldsymbol{y}(s)$ [14] is

$$\log \mathrm{p}_T(X(s) \mid \boldsymbol{y}(s)) = G(s) + H_T(s, \boldsymbol{y}(s)) \tag{A.5}$$

where $G(s) = \sum_{c=1}^{C} N_c(s) \log \left((2\pi)^{-\frac{F}{2}} \left|\Sigma_c\right|^{-\frac{1}{2}} \right) - \frac{1}{2} \operatorname{tr} \left(\Sigma^{-1} S_{XX}(s) \right)$ is a term independent of T and $\boldsymbol{y}(s)$. Hence it will not play a role in estimating T and $\boldsymbol{y}(s)$. The second term is

$$H_T(s, \boldsymbol{y}(s)) \;=\; \boldsymbol{y}(s)^{\intercal} T^{\intercal} \Sigma^{-1} \boldsymbol{F}_X(s) - \frac{1}{2} \boldsymbol{y}(s)^{\intercal} T^{\intercal} \Sigma^{-1} N(s) T \boldsymbol{y}(s) \tag{A.6}$$

It can be shown that the *aposteriori* density function of $\boldsymbol{y}(s)$ given $X(s)$ is Gaussian with covariance $\mathcal{L}(s)^{-1}$ and mean $\mathcal{L}(s)^{-1} T^{\intercal} \Sigma^{-1} \boldsymbol{F}_X(s)$ [14] where

$$\mathcal{L}(s) = I + T^{\intercal} \Sigma^{-1} N(s) T \tag{A.7}$$

Estimating T by maximum likelihood is done by using the Expectation-Maximization algorithm.

Appendix A.1. Expectation (E) step:
The Q function is given by:

$$\begin{aligned} Q\left(T \mid T^{(t)}\right) &= \sum_{s=1}^{S} \mathbb{E}_{\boldsymbol{y}(s)|X(s),T^{(t)}} \log \mathrm{p}_T(X(s), \boldsymbol{y}(s)) \\ &= \sum_{s=1}^{S} \mathbb{E}_{\boldsymbol{y}(s)|X(s),T^{(t)}} \log \mathrm{p}_T(X(s) \mid \boldsymbol{y}(s)) \\ &\qquad + \sum_{s=1}^{S} \mathbb{E}_{\boldsymbol{y}(s)|X(s),T^{(t)}} \log \mathrm{p}(\boldsymbol{y}(s)) \end{aligned} \tag{A.8}$$

The second term is independent of T and doesn't play a role in the M-Step, and hence it can be ignored. Substituting for the likelihood term from Eq (A.5), ignoring the irrelevant terms and simplifying yields:

$$\begin{aligned} Q\left(T \mid T^{(t)}\right) = \sum_{s=1}^{S} \Bigg[& \operatorname{tr} \left\{ T^{\intercal} \Sigma^{-1} \boldsymbol{F}_X(s) \hat{\boldsymbol{y}}^{(t)}(s)^{\intercal} \right\} \\ & - \frac{1}{2} \operatorname{tr} \left\{ \Sigma^{-1} N(s) T E_{yy}^{(t)}(s) T^{\intercal} \right\} \Bigg] \end{aligned} \tag{A.9}$$

where

$$\mathcal{L}^{(t)}(s) = I + T^{(t)\intercal}\Sigma^{-1}N(s)T^{(t)} \tag{A.10}$$

$$\hat{\boldsymbol{y}}^{(t)}(s) = \mathcal{L}^{(t)}(s)^{-1}T^{(t)\intercal}\Sigma^{-1}\boldsymbol{F}_X(s) \tag{A.11}$$

$$E^{(t)}_{yy}(s) = \mathcal{L}^{(t)}(s)^{-1} + \hat{\boldsymbol{y}}^{(t)}(s)\hat{\boldsymbol{y}}^{(t)}(s)^{\intercal} \tag{A.12}$$

The E-step can be summarized as computing the quantities in Eq (A.10 - A.12). The vector $\hat{\boldsymbol{y}}^{(t)}(s)$ is the MAP estimate of the latent variable computed using the total variability matrix $T^{(t)}$ which is usually as the i-vector.

Appendix A.2. Maximization (M) step:

The update equation for the matrix T is obtained by maximizing the Q function w.r.t T.

$$T^{(t+1)} = \underset{T}{\operatorname{argmax}}\, Q\left(T \mid T^{(t)}\right) \tag{A.13}$$

Differentiating Eq (A.9), equating it to zero and simplifying yields the system of linear equations in every row of $T^{(t+1)}$:

$$\sum_{s=1}^{S} N(s)T^{(t+1)}\left(\mathcal{L}^{(t)}(s)^{-1} + \hat{\boldsymbol{y}}^{(t)}(s)\hat{\boldsymbol{y}}^{(t)}(s)^{\intercal}\right) = \sum_{s=1}^{S}\boldsymbol{F}_X(s)\hat{\boldsymbol{y}}^{(t)}(s)^{\intercal} \tag{A.14}$$

$T^{(t+1)}$ is obtained by solving the above systems of linear equations.

Appendix B. Expectation Maximization algorithm for estimating the parameters of the s-vector model

The class conditioned means $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_L$ can be parameterized along with the matrix T. We define the new parameter set as $\Theta = \{T, \boldsymbol{m}_1, \ldots, \boldsymbol{m}_L\}$

The likelihood function of Θ in terms of $X(s), l(s)$ and $\boldsymbol{y}(s)$ is

$$\begin{aligned}\mathrm{p}_\Theta(X(s), \boldsymbol{y}(s), l(s)) &= \mathrm{p}_\Theta(X(s), \boldsymbol{y}(s) \mid l(s))\mathrm{p}(l(s)) \\ &= \frac{1}{L}\mathrm{p}_\Theta(X(s) \mid \boldsymbol{y}(s))\mathrm{p}_\Theta(\boldsymbol{y}(s) \mid l(s))\end{aligned} \tag{B.1}$$

The complete data log likelihood is

$$\begin{aligned}&\sum_{s=1}^{S}\log\mathrm{p}_\Theta(X(s), \boldsymbol{y}(s), l(s)) \\ &\qquad = S\log\frac{1}{L} + \sum_{s=1}^{S}\log\mathrm{p}_\Theta(X(s) \mid \boldsymbol{y}(s)) + \sum_{s=1}^{S}\log\mathrm{p}_\Theta(\boldsymbol{y}(s) \mid l(s))\end{aligned} \tag{B.2}$$

It can be shown that the *aposteriori* density function of $\boldsymbol{y}(s)$ given $X(s)$ and $l(s)$ is Gaussian with covariance $\mathcal{L}^{-1}(s)$ which is the same as Eq (A.7) and mean given by $\mathcal{L}^{-1}(s)\big\{\boldsymbol{m}_{l(s)} + T^{\intercal}\Sigma^{-1}\boldsymbol{F}_X(s)\big\}$.

The EM algorithm (supervised EM) is used to solve for the parameters $\Theta = \{T, \boldsymbol{m}_1, \ldots, \boldsymbol{m}_L\}$ which maximize the joint likelihood function $\sum_{s=1}^{S} \log \mathrm{p}_\Theta(X(s), l(s))$.

Appendix B.1. Expectation (E) Step:

The Q function in terms of $X(s)$ and $l(s)$ is given by

$$\begin{aligned} Q\left(\Theta \mid \Theta^{(t)}\right) &= \sum_{s=1}^{S} \mathbb{E}_{\boldsymbol{y}(s)|X(s),l(s),\Theta^{(t)}} \log \mathrm{p}_\Theta(X(s), l(s), \boldsymbol{y}(s)) \\ &= \sum_{s=1}^{S} \mathbb{E}_{\boldsymbol{y}(s)|X(s),\Theta^{(t)}} \log \mathrm{p}_\Theta(X(s) \mid \boldsymbol{y}(s)) \\ &\qquad + \sum_{s=1}^{S} \mathbb{E}_{\boldsymbol{y}(s)|X(s),\Theta^{(t)}} \log \mathrm{p}(\boldsymbol{y}(s) \mid l(s)) \end{aligned} \tag{B.3}$$

Here, we have used the independence assumption (Eq 5). Substituting the required terms from Eq (A.5, 4 & B.2) and simplifying gives

$$\begin{aligned} Q\left(\Theta|\Theta^{(t)}\right) = \sum_{s=1}^{S} \left[\mathrm{tr}\left\{ T^{\intercal}\Sigma^{-1}\boldsymbol{F}_X(s)\hat{\boldsymbol{y}}^{(t)}_{l(s)}(s)^{\intercal} \right\} - \frac{1}{2}\mathrm{tr}\left\{ \Sigma^{-1}N(s)TE^{(t)}_{yy,l(s)}(s)T^{\intercal} \right\} \right] \\ + \sum_{l=1}^{L} \sum_{\substack{s=1 \\ l(s)=l}}^{S} \left(\hat{\boldsymbol{y}}^{(t)}_{l(s)}(s)^{\intercal}\boldsymbol{m}_l - \frac{1}{2}\boldsymbol{m}_l{}^{\intercal}\boldsymbol{m}_l \right) \end{aligned} \tag{B.4}$$

where $\mathcal{L}^{(t)}(s)$ is the same as given in Eq (A.10) and

$$\hat{\boldsymbol{y}}^{(t)}_{l(s)}(s) = \mathcal{L}^{(t)}(s)^{-1}\left\{ \boldsymbol{m}_{l(s)} + T^{(t)\intercal}\Sigma^{-1}\boldsymbol{F}_X(s) \right\} \tag{B.5}$$

$$E^{(t)}_{yy,l(s)}(s) = \mathcal{L}^{(t)}(s)^{-1} + \hat{\boldsymbol{y}}^{(t)}_{l(s)}(s)\hat{\boldsymbol{y}}^{(t)}_{l(s)}(s)^{\intercal} \tag{B.6}$$

Appendix B.2. Maximization (M) step:

The update equation for the matrix T is same as in Eq (A.14), and the update equation for $\boldsymbol{m}_l$ can be obtained by partially maximizing the Q

function w.r.t $\boldsymbol{m}_l$ and is given by

$$\boldsymbol{m}_l^{(t+1)} = \frac{1}{S_l} \sum_{\substack{s=1 \\ l(s)=l}}^{S} \hat{\boldsymbol{y}}_{l(s)}^{(t)}(s) \tag{B.7}$$

where S_l is the number of utterances having class label l. It can be noted that $S = \sum_{l=1}^{L} S_l$.

Appendix B.3. Minimum Divergence Re-estimation

In the traditional (unsupervised) EM algorithm, Minimum divergence re-estimation of the T matrix is done after every M-step to ensure that the model is consistent with the prior on the latent variable $\boldsymbol{y}$. Assume that the current estimate T results in i-vector estimates with covariance $K_{\boldsymbol{yy}}$ instead of identity covariance, the TVM equation can be written as

$$\boldsymbol{M}(s) = \boldsymbol{M}_0 + T\boldsymbol{y}(s) \tag{B.8}$$

$$= \boldsymbol{M}_0 + \underbrace{TK_{\boldsymbol{yy}}^{\frac{1}{2}}}_{T'} \underbrace{\left(K_{\boldsymbol{yy}}^{-\frac{1}{2}}\boldsymbol{y}(s)\right)}_{\boldsymbol{y}'(s)} \tag{B.9}$$

Hence, by making the modifications $T' \leftarrow TK_{\boldsymbol{yy}}^{\frac{1}{2}}$ and $\boldsymbol{y}'(s) \leftarrow K_{\boldsymbol{yy}}^{-\frac{1}{2}}\boldsymbol{y}(s)$, and re-estimating $\{\boldsymbol{m}_l\}_{l=1}^{L}$ using $\boldsymbol{y}'(s)$, the KL divergence of the distribution of i-vectors using updated model from the prior on $\boldsymbol{y}(s)$ will be minimized. The quantity $K_{\boldsymbol{yy}}$ is computed as

$$K_{\boldsymbol{yy}} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{E}_{\boldsymbol{y}(s)|X(s),T} \left[\boldsymbol{y}(s)\boldsymbol{y}(s)^{\intercal}\right] \tag{B.10}$$

It can be shown that by making these updates, the total data log likelihood also improves [13].

$$\sum_{s=1}^{S} \log \mathrm{p}_{T'}(X(s)) \geq \sum_{s=1}^{S} \log \mathrm{p}_{T}(X(s)) \tag{B.11}$$

If the mean of the i-vectors is assumed to be $\boldsymbol{m}$ instead of zero mean, $K_{\boldsymbol{yy}}$ must be computed as

$$K_{\boldsymbol{yy}} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{E}_{\boldsymbol{y}(s)|X(s),T} \left[(\boldsymbol{y}(s) - \boldsymbol{m})(\boldsymbol{y}(s) - \boldsymbol{m})^{\intercal}\right] \tag{B.12}$$

For the s-vector model, we have modeled the class conditioned prior on $\boldsymbol{y}(s)$ for each class l to be Gaussian with mean $\boldsymbol{m}_l$ and identity covariance matrix shared by all classes. Hence, the covariance normalization done in the earlier can be replaced by within class covariance normalization. That is, $K_{\boldsymbol{yy}}$ should be the within class covariance of $\boldsymbol{y}$ and is computed as

$$K_{\boldsymbol{yy}} = \frac{1}{S}\sum_{s=1}^{S} \mathbb{E}_{\boldsymbol{y}(s)|X(s),l(s),T}\left[(\boldsymbol{y}(s)-\boldsymbol{m}_{l(s)})(\boldsymbol{y}(s)-\boldsymbol{m}_{l(s)})^{\intercal}\right] \tag{B.13}$$

Expanding and simplifying using the parameters at time step t,

$$K_{yy}^{(t)} = \frac{1}{S}\sum_{s=1}^{S}\Big(E_{yy,l(s)}^{(t)}(s) + \hat{\boldsymbol{y}}_{l(s)}^{(t)}(s)\hat{\boldsymbol{y}}_{l(s)}^{(t)}(s)^{\intercal} - \hat{\boldsymbol{y}}_{l(s)}^{(t)}(s)\boldsymbol{m}_{l(s)}^{(t)\ \intercal} - \boldsymbol{m}_{l(s)}^{(t)}\hat{\boldsymbol{y}}_{l(s)}^{(t)}(s)^{\intercal} + \boldsymbol{m}_{l(s)}^{(t)}\boldsymbol{m}_{l(s)}^{(t)\ \intercal}\Big) \tag{B.14}$$

Appendix C. Computing label posteriors for s-vector extraction

The MMSE s-vectors are extracted using the expression in Eq. (13). Here, we explain how is the quantity $\mathrm{p}(l \mid X(s))$ computed.

The log likelihood of $X(s)$ for class l can be written as

$$\log \mathrm{p}(X(s) \mid l) = \log \int_{<\boldsymbol{y}(s)>} \mathrm{p}(X(s), \boldsymbol{y}(s) \mid l(s))\ \mathrm{d}\boldsymbol{y}(s) \tag{C.1}$$

$$= \log \int_{<\boldsymbol{y}(s)>} \mathrm{p}(X(s) \mid \boldsymbol{y}(s), l(s))\ \mathrm{p}(\boldsymbol{y}(s) \mid l(s))\ \mathrm{d}\boldsymbol{y}(s) \tag{C.2}$$

$$= \log \int_{<\boldsymbol{y}(s)>} \mathrm{p}(X(s) \mid \boldsymbol{y}(s))\ \mathrm{p}(\boldsymbol{y}(s) \mid l(s))\ \mathrm{d}\boldsymbol{y}(s) \tag{C.3}$$

The $l(s)$ term vanishes due to the independence assumption (Eq 5). Substituting for the log likelihood from Eq. (A.5) and the expression for the class conditioned prior of $\boldsymbol{y}(s)$ with reweighting factor λ, we get

$$\log \mathrm{p}(X(s) \mid l) = \log \int_{<\boldsymbol{y}(s)>} e^{(G(s)+H_T(s,\boldsymbol{y}(s))}\frac{1}{(2\pi)^{\frac{R}{2}}}e^{-\frac{1}{2}\lambda(\boldsymbol{y}(s)-\boldsymbol{m}_l)^{\intercal}(\boldsymbol{y}(s)-\boldsymbol{m}_l)}\mathrm{d}\boldsymbol{y}(s) \tag{C.4}$$

Simplifying, we get

$$\log \mathrm{p}(X(s) \mid l) = G(s) - \frac{1}{2}\lambda \boldsymbol{m}_l{}^{\intercal}\boldsymbol{m}_l - \frac{1}{2}\log|\mathcal{L}(s)| + \frac{1}{2}(\lambda \boldsymbol{m}_l + T^{\intercal}\Sigma^{-1}\boldsymbol{F}_X(s))^{\intercal}\mathcal{L}(s)^{-1}(\lambda \boldsymbol{m}_l + T^{\intercal}\Sigma^{-1}\boldsymbol{F}_X(s)) \tag{C.5}$$

The class posteriors $\mathrm{p}(l|X(s))$ can be calculated using the Bayes' formula, and terms $G(s)$ and $-\frac{1}{2}\log|\mathcal{L}(s)|$ are independent of l and can be ignored while calculating the posteriors.