An Investigation of Non-linear i-vectors for speaker verification

Nanxin Chen, Jesús Villalba, Najim Dehak

Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD
{bobchennan,jvillal7, ndehak3}@jhu.edu

Abstract

Speaker verification becomes increasingly important due to the popularity of speech assistants and smart home. i-vectors are used broadly for this topic, which use factor analysis to model the shift of average parameter in Gaussian Mixture Models. Recently by the progress of deep learning, high-level non-linearity improves results in many areas. In this paper we proposed a new framework of i-vectors which uses stochastic gradient descent to solve the problem of i-vectors. From our preliminary results stochastic gradient descent can get same performance as expectation-maximization algorithm. However, by backpropagation the assumption can be more flexible, so both linear and non-linear assumption is possible in our framework. From our result, both maximum a posteriori estimation and maximum likelihood lead to slightly better result than conventional i-vectors and both linear and non-linear system has similar performance.

Index Terms: speaker verification, generative model, stochastic gradient descent

1. Introduction

With the increasing popularity of smart phones and home assistants, speech related techniques draw more and more attention from both academic and industry. While speech recognition performance keeps improving, now people start to concern more about privacy issues. Given the fact that personal information is usually shared with those hardware, one natural question is how we can make it secure. Speaker verification becomes a natural choice and becomes common on many home assistants. Speaker verification systems distinguish different speakers by their physiological characteristics, and reply to the corresponding user.

Speaker verification research has a very long history. Started from 20th century, Gaussian Mixture Model Universal Background Model (GMM-UBM) was proposed in [1] achieving great performance. The UBM model is trained on large corpus to model the speaker independent speech features distribution. Then, Maximum a posteriori is used to adapt the GMM parameters to any specific speaker. Later, joint factor analysis (JFA) [2, 3] enhanced the performance of GMM-UBM system. JFA makes the assumption that between and within speaker variability spaces are of low-rank. While JFA assumes different factors for channel and speaker, following research shows that even channel factors have information related to speaker and can improve overall performance. In 2011, Dehak et al. [4] proposed a simplified framework, which is broadly known as i-vectors, used single factor defined by a total variability matrix. The point estimate of the total variability factor (i-vector) is used as a new feature for other classifiers, e.g., probabilistic linear discriminant analysis (PLDA) [5] is used to make same/different speaker decisions[6, 7].

Recently, with the advent of deep learning technologies, more research [8, 9, 10, 11, 12, 13, 14, 15, 16] turned to use neural network to model the speaker variability. In [8, 9], neural network is used to replace GMM in GMM-UBM model to provide alignment and statistics. In [11], bottleneck features are extracted from neural network to make the GMM to get better unsupervised clustering of the feature space improving results. Later, different works are proposed to directly use the neural network for speaker verification [10, 12, 13, 14, 15]. These networks require to be trained on large amount of data, usually using supervise learning, and show great performance on both text-dependent[10, 12, 13, 14, 15] and text-independent[16] speaker verification.

In this paper we proposed a new framework combining conventional i-vectors with neural network. More specifically, a generative model that generalizes i-vectors is proposed and we show that i-vectors is a special case of our new framework. Back-propagation based stochastic gradient descent is used to train the model parameters and infer the latent factors. While representing i-vectors in this new framework, we are enabled to add non-linearity to i-vectors. Our preliminary results on SRE16 show that

- minimum divergence estimation is helpful for inference
- Performance of linear and non-linear systems are similar
- Both maximum a posteriori and maximum likelihood can slightly improve results of conventional i-vectors

This paper is organized as follows. Section 2 introduces conventional i-vector systems and the proposed new i-vector framework. Section 3 reports result for i-vector baseline system and explores different factors: minimum divergence estimation, non-linearity, maximum likelihood estimation and maximum a posteriori estimation. Section 4 summarizes the paper and gives directions for future research.

2. i-Vectors and Non-linear i-vectors

2.1. i-vectors

The i-vector framework models the utterance features as a Gaussian mixture model (GMM). The shift of utterance dependent GMM supervector mean w.r.t. the universal background model is assumed to be

\[ m = M + Tw \]  

(1)

where \( M \) is the original UBM mean, and \( T \) is a low-rank matrix mapping low-dimensional i-vector to the high-dimensional supervector space. The supervector is the concatenation of all mean parameters. \( T \) is usually called total variability matrix.

Expectation maximization is used to train both UBM and i-vectors model parameters. After extracting i-vectors as fixed-dimensional embedding, both cosine similarity and probabilistic linear discriminant analysis [5] can be used.
2.2. Non-linear i-vectors

2.2.1. Model definition

In this work as in the i-vector framework, we assumed that the utterance features are generated by a GMM,

\begin{align}
z_{ij} & \sim \text{Categorical}(\phi) \\
x_{ij} & \sim \mathcal{N}(x_{ij} | M_{x_{ij}} + G_{x_{ij}}(w_i; \theta), \Sigma_{x_{ij}}) \tag{2}
\end{align}

where \( x_{ij} \) is frame \( j \) from utterance \( i \), \( z_{ij} \) is one-hot vector of Gaussian occupations, and \( \phi \) are the mixture weights. \( G \) is general function that maps the latent factor \( w \) to the shift between the utterance dependent supervector mean and UBM mean.

Conventional i-vectors can be considered as

\[ G^i_w(w; \theta) = T_{\theta z_i} w \tag{3} \]

which is a special case of Equation (2). However, in this work we try to add non-linearity to the function \( G \). One natural idea is to add activation function directly to the product of \( T_{\theta} \) and \( w \):

\[ G^i_w(w; \theta) = g(T_{\theta z_i} w) \tag{4} \]

As in the i-vector framework, we will assume that \( z_{ij} \) posteriors are given by the GMM-UBM.

2.2.2. Theory

Theory behind proposed framework can be more general. We want to model connections between observed variables \( X \)–the feature frames of a given utterance– and a latent variable \( w \). In a generative setting, we want to maximize the marginal likelihood of the observed data \( X \) given the model parameters \( \theta \), \( \log p(X|\theta) \). However in practice, this is intractable in most cases. Thus, we maximize a lower bound (ELBO) for that likelihood instead,

\[ L(X, \theta, q) = \log p(X|\theta) - D_{K-L}(q(w|X)||p(w|X, \theta)) \tag{5} \]

where \( q \) is any distribution over \( w \). Because KL-divergence is non-negative, \( L(X, \theta, q) \) becomes a lower bound of \( \log p(X|\theta) \). It is actually easier to compute since it can be rearranged as

\[ L(X, \theta, q) = -E_{q(w|X)}(\log q(w|X) - \log p(w, X|\theta)) \tag{6} \]

Variational autoencoder is developed under this theory where encoder models \( q(w|x) \) and decoder models \( p(x|w) \). For embedding extraction we care more about \( q(w|X) \). In this work, we simplify this model and we just want to make a point estimate \( \mu \) of the latent factor. For that, we assume that the form of \( q \) is a Dirac delta distribution,

\[ q(w|X) = \delta(w - \mu) \tag{7} \]

where \( \mu \) is the latent factor point estimate. Thus, Equation (6) can be simplified as

\[ L(w, \theta, q = \delta(w - \mu)) = H(q) + \log p(w = \mu, X|\theta) = \log p(X|w = \mu; \theta) + \log p(w = \mu) \tag{8} \]

where \( H(q) \) is the entropy of distribution \( q \). So maximizing \( L(X, \theta, q = \delta(w - \mu)) \) derives maximum a posteriori inference [17].

![Figure 1: Diagram of our model. \( W, M, V \) is weight, mean and co-variance of UBM, respectively. Response is fixed thus there is no backward pass from response to UBM.](image)

In our case, we care about how to find the optimal \( \mu \). Modeling the latent posterior \( p(w|X) \) is difficult, since \( X \) contains a variable number of frames, so we need to devise some kind of complex mechanism to accumulate the information of all the frames into the unique latent factor \( w \). However, similar to sparse coding, we treated this as an optimization problem, which simplified the problem. Instead of using closed form function to obtain \( \mu \), as it is proposed in the variational autoencoder framework [18], we proposed to assume that \( \mu \) is a trainable parameter that we obtain by back-propagation. In this manner, no encoder function is required but just the decoder distribution \( p(X|w) \). GMM-UBM is adopted as decoder in order to compare with i-vectors. The same UBM is trained on same datasets and used for both i-vectors and our method.

The diagram of whole process is given in Figure 1. In each mini-batch, we calculate response of each Gaussian based on the UBM model. Then all the embeddings are gathered according to data \( X \) in the mini-batch and update the supervector of UBM by function \( G \). Loss value is calculated by data \( X \) and updated UBM. In backward, starting from loss value, gradient of \( \theta \) and \( w \) is calculated by recursion. In each mini-batch, we updated \( \theta \) and corresponding \( w \) in this mini-batch. In inference (i-vector extraction), only \( \mu \) is updated while \( \theta \) is fixed.

Although our approach is minibatch-based training, we want to emphasize the similarities between EM algorithm and SGD. SGD in general can be seen as a special case of EM. In each mini-batch we optimize \( w \) given \( \theta \) in previous batch and we optimize \( \theta \) given last \( w \). Even each mini-batch cannot provide accurate gradient information and the step size is relatively small, with enough updates the system should converge to a similar point.

2.2.3. Loss function

In this work we consider two loss function according to Equation (8):

88
Table 1: Training data details: number of utterances from each corpus and proportion of gender

<table>
<thead>
<tr>
<th>Corpus</th>
<th>#utts</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRE05</td>
<td>1165</td>
<td>38%</td>
<td>62%</td>
</tr>
<tr>
<td>SRE06</td>
<td>5245</td>
<td>45%</td>
<td>55%</td>
</tr>
<tr>
<td>SRE08</td>
<td>4504</td>
<td>39%</td>
<td>61%</td>
</tr>
<tr>
<td>swb1</td>
<td>30972</td>
<td>46%</td>
<td>54%</td>
</tr>
</tbody>
</table>

Two configurations are tested in our experiments:

\[ G_1(w) = M + g_1(T_{θw}) \]
\[ G_2(w) = M + g_2(T_{θw}) \]

(11)

In the following we introduce the method used to create the mini-batches that we used in our stochastic back-propagation algorithm. Because of the independence assumption, we can shuffle the frames in each utterance without altering the model. By shuffling the frames, we increase the variability of the feature chunks that we used to train the model. It is known that increasing randomness it is beneficial to make deep models to generalize well. After that we split each utterance into small chunks where each chunk included 128 frames. All those chunks were shuffled in each epoch before feeding them into our model. Each mini-batch contained 200 chunks. From our observations, both size of chunk and number of chunks in each mini-batch is quite important for final performance. The first one decides how accurate gradient of \( w \) is and the second one decides how much inter-session variability there is in the mini-batch. In the training learning rate is set to 0.001 and Adam[22] was used. For inference learning rate is set to 0.005 and 10 iterations are used to find the optimal \( w \). For fair comparison, we also set \( w \) to 400-dimensional embedding. For both i-vectors and non-linear i-vectors, probabilistic linear discriminant analysis (PLDA) was adopted as the back-end.

3. Experiments

3.1. Datasets

We experimented using the setup, that we used in our participation in NIST SRE16 evaluation [19, 20]. In this setup, the training set included two parts:

- swb1: Switchboard 1 data with each original cut segmented to a single 10 - 60 seconds. The duration is drawn form a uniform distribution
- sre568eng: 2005-2008 NIST English SRE data also uniformly segmented from 10 - 60 seconds.

There were 41859 utterances in total. More details of data is included in Table 1.

All frameworks were tested on Cantonese portion of the NIST SRE 2016 evaluation (SRE16). Enrollment utterances contained about 60 seconds of conversational telephone speech while the test utterances varied from 10 - 60 seconds. We estimated mean of i-vectors from NIST SRE16 development set “major” data and use it to center all i-vectors for enrollment and test data.

3.2. Experiments setup

UBM model was trained on whole training set and diagonal co-variance was used in all experiments. Both UBM model and i-vectors baseline were trained using Kaldi [21].

Results for 400-dimensional i-vectors baseline with different number of diagonal Gaussians using MAP estimation is presented in Table 2. From the table, clearly more Gaussians give better result but the difference is not that large. Due to GPU memory limit we used the 256 Gaussians UBM to test our approach.

Table 2: 400-dimensional i-vectors system baseline

<table>
<thead>
<tr>
<th>#Gaussians</th>
<th>EER</th>
<th>DCF10^-2</th>
<th>DCF10^-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDE</td>
<td>0.005</td>
<td>13.18</td>
<td>0.722</td>
</tr>
<tr>
<td>no MDE</td>
<td>0.005</td>
<td>14.05</td>
<td>0.746</td>
</tr>
<tr>
<td>no MDE</td>
<td>0.0005</td>
<td>13.21</td>
<td>0.723</td>
</tr>
</tbody>
</table>

Table 3: Effect of minimum diverge estimation (MDE). With minimum diverge estimation fixed learning rate can be used. Otherwise adjusting learning rate manually is needed.

3.3. Results

3.3.1. Effect of minimum diverge estimation

As discussed in Section 2.2.3, minimum diverge estimation is used to reduce the effect of imbalance between scale of \( T_θ \) and variance of \( w \). Results are given in Table 3.

Clearly from table minimum diverge estimation is helpful and even learning rate can be manually set to very small number (like 0.0005 in the table), minimum diverge estimation still improves the result. This is coherent with what we know from the standard i-vector practice.

3.3.2. Comparison between linearity and non-linearity

For non-linearity parametric rectified linear unit[23] is adopted as function \( g_\alpha \) in Equation (4):

\[ g_\alpha(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{otherwise} \end{cases} \]

(10)

where \( \alpha \) is a parameter learned from data and fixed for all dimensions. During training we start \( \alpha \) from 1, which correspond to linear case.

Two configurations are tested in our experiments:

\[ G_1(w) = M + g_1(T_{θw}) \]
\[ G_2(w) = M + g_1(T_1 g_2(T_2 w)) \]

(11)
where \( T_2 \) project \( w \) into an intermediate hidden space with 1024 dimension. \( g_2 \) has different parameters.

Results in Table 4 reveals that non-linearity doesn’t give improvement. \( G_1 \) and \( G_2 \) is named as 1 Layer PReLU and 2 Layer PReLU, respectively. We find out that \( \alpha \) for \( g_2 \) is close to 0 which makes intermediate representation extremely sparse. Result suggests linear assumption is good enough for mean shift compared to preliminary non-linear assumption. For non-linear case more further research is still needed.

### 3.3.3 Comparison between maximum likelihood estimation estimation \( \text{MLE} \) and maximum a posteriori \( \text{MAP} \) estimation

The only difference between MLE and MAP estimation is the weight of prior in Equation (8). Table 5 reports results of MLE and MAP. From the table it is obvious that MAP estimation has similar performance as ML estimation. However, with the introduction of prior term \( \mu(w = \mu) \), \( w \) is closer to standard normal distribution, which reduces the relative improvement of minimum divergence estimation. We further plot distribution of randomly selected dimension in training for both MLE and MAP and plot in Figure 2. From figure \( w \) tends to have similar distribution as standard normal in MAP, compared to MLE. Overall, maximum likelihood estimation gives better decision cost function (DCF) while maximum a posteriori estimation lead to better equal error rate (ERR).

### 4. Conclusions

In this paper we presented a novel stochastic based non-linear i-vectors system. The proposed model follows the UBM approach, the same as i-vector. However, it generalizes i-vectors allowing to use a non-linear function to map the latent factor (i-vector) to the utterance dependent GMM super-vector mean. In this kind of model estimating the latent factor or the model parameters is intractable. However, we can resort to a similar approach used in variational autoencoders [18], where model parameters are obtained by stochastic back-propagation and latent factor are computed by a deep neural network. In our framework, we proposed to use stochastic back-propagation to compute both latent factors and model parameters. In the training phase, model parameters and latent factors were estimated iteratively. Meanwhile in evaluation phase—when obtaining enrollment and test i-vectors—, the model parameters were fixed.

We experimented with this framework using linear and non-linear decoders to compute the GMM means. Our experiments on SRE16 show that our framework with both MLE and MAP training lead to slightly better result than conventional i-vectors. Further work, includes extending non-linearity by allowing different non-linearity in each dimension, and adapting different parameters simultaneously.

### 5. References


