BAT System Description for NIST LRE 2015

I–vector transformation and scaling for PLDA based speaker recognition

Sandro Cumani and Pietro Laface

{Sandro.Cumani, Pietro.Laface}@polito.it

Abstract

This paper proposes a density model transformation for speaker recognition systems based on i–vectors and Probabilistic Linear Discriminant Analysis (PLDA) classification. The PLDA model assumes that the i-vectors are distributed according to the standard normal distribution, whereas it is well known that this is not the case. Experiments have shown that the i–vector are better modeled, for example, by an Heavy–Tailed distribution, and that significant improvement of the classification performance can be obtained by whitening and length normalizing the i-vectors. In this work we propose to transform the i–vectors, extracted ignoring the classifier that will be used, so that their distribution becomes more suitable to discriminate speakers using PLDA. This is performed by means of a sequence of affine and non–linear transformations, the parameters of which are obtained by Maximum Likelihood (ML) estimation on the training set. The second contribution of this work is the reduction of the mismatch between the development and test i–vector distributions by means of a scaling factor tuned for the estimated i–vector distribution, rather than by means of a blind length normalization. Our tests performed on the NIST SRE-2010 and SRE-2012 evaluation sets show that improvement of their Cost Functions of the order of 10% can be obtained for both evaluation data.

1. Introduction

Systems based on i–vectors [1] and on Probabilistic Linear Discriminant Analysis (PLDA) [2, 3, 4], or discriminative classifiers [5], represent the current state–of–the–art in text–independent speaker recognition. The i–vector is a compact representation of a speech segment, obtained from the statistics of a Gaussian Mixture Model (GMM) supervector [6] by a Maximum a Posteriori point estimate of a posterior distribution [1].

It has been shown that better i–vectors can be obtained by means of hybrid DNN/GMM architectures that may take advantage of the information that is not exploited by the traditional GMM approach: the phonetic content of a rather large window of frames [7, 8]. In particular, in this approach, a fine–grained 'phonetic' Universal Background Model (UBM) is obtained by associating one or more Gaussians [9] to each output unit of a DNN, trained to discriminate among the states of a set of context–dependent phonetic units. Another approach for obtaining better i–vectors exploits DNN bottleneck features, derived form the input of a layer with a small number of units located in the middle of a DNN architecture. These features have been used as a replacement of, or in combination with, the standard MFCC features, showing good performance improvement [10, 11, 12] in text–independent and also in text–dependent speaker recognition [13].

Computational resources for this work were provided by HPC@POLITO (http://www.hpc.polito.it) Politecnico di Torino, Italy

All these methods of i–vector extraction are performed ignoring the model that will be used for classification, but better results are expected if the features provided to a classifier fulfill its assumptions. In this work we focus on an i–vector post–processing technique that allows obtaining better features for a PLDA classifier.

A PLDA classifier models the underlying distribution of the speaker and channel components of the i–vectors in a probabilistic framework. From these distributions it is possible to evaluate the likelihood ratio between the "same speaker" hypothesis and "different speaker" hypothesis for a pair of i–vectors. In particular, PLDA assumes that the i–vector generation process can be described by means of a latent variable probabilistic model where an i–vector $\boldsymbol{\phi}$ is modeled as the sum of three factors, namely a speaker factor $\mathbf{y}$, an inter–session (channel) factor $\mathbf{x}$ and the residual noise ϵ as:

$$\boldsymbol{\phi} = \mathbf{U}_1\mathbf{y} + \mathbf{U}_2\mathbf{x} + \epsilon\,.$$

Matrices $\mathbf{U}_1$ and $\mathbf{U}_2$ typically constrain the speaker and inter–session factors to be of lower dimension than the i–vectors space. PLDA estimates the matrices $\mathbf{U}_1$, $\mathbf{U}_2$, and the values of the hyper–parameters of possible parametric priors [2], which maximize the likelihood of the observed i–vectors, assuming that i–vectors from the same speaker share the same speaker factor, i.e., the same value for latent variable $\mathbf{y}$.

The simplest PLDA model (G–PLDA) assumes also a Gaussian distribution for the latent variables and i–vectors. However, in [2] it has been shown that ML estimation of the PLDA parameters under Gaussian assumption fails to produce accurate models for i-vectors. Thus, heavy–tailed distributions for the model priors have been proposed leading to the Heavy-Tailed PLDA model. This model, however, is computationally expensive both in training and in testing, thus will be not considered in our experiments.

A simpler approach has been proposed in [14] that tries to make more Gaussian-like the distribution of the i–vectors. It incorporates a pre–processing step where the vector dimensionality is possibly further reduced by Linear Discriminant Analysis (LDA), and, what is more important, length normalization (LN) is applied to the resulting features. Using these normalized i–vectors, the performance of the Heavy–Tailed and Gaussian PLDA models is comparable, the latter being much faster both in training and in testing. A similar approach was also proposed in [15]. Another technique to better fit the assumption of PLDA that the i–vectors are Gaussian distributed is the Spherical Nuisance normalization applied to the development and test i–vectors [16].

It is worth noting that LN aims at reducing both the non-Gaussian behavior of the i–vector, and the mismatch between the development and test i-vector length distributions. However, as well documented in [14], keeping the development i-vectors in their original form does not affect the performance once LN has been applied to the test data. This suggests that LN mostly

compensates the mismatch between the development and test i–vector length distributions, rather than obtaining a transformation of the i–vectors that fits a standard normal distribution.

In this work we cope with both problems:

- i–vectors are transformed so that their distribution becomes more Gaussian–like by means of a sequence of affine and non–linear transformations, the parameters of which are obtained by Maximum Likelihood (ML) estimation on the development set.
- the mismatch between the development and test i-vector length distributions is reduced by estimating an i–vector dependent scaling factor.

We show that this approach, due to the gaussianization of the i–vectors, is able to improve the performance of a PLDA classifier when LN is not used, and to produce better results, compared to the standard G–PLDA with LN, when the density transformation is performed in conjunction with either LN or scaling–factor normalization.

The paper is organized as follows: Section 2 introduces and analyzes the density model transformations. The i–vector post–processing transformations that we have used is illustrated in Section 3. Section 4 presents the proposed scaling–factor normalization technique. Sections 5 and 6 are devoted to the experimental settings and results, respectively, and conclusions are drawn in Section 7.

2. Density function transformations

Since we are interested in mapping a set of i–vectors so that their (unknown) distribution becomes Gaussian–like, we are facing the problem that given two probability density functions (pdf), we need a function which is able to transform one into the other. We can cast this problem as estimating the pdf of a random variable whose distribution is unknown by means of ML estimation of a parametric transformation of a random variable with known pdf.

Let function

$$\begin{aligned} f : S_1 \times Q &\to S_2 \\ (\mathbf{x}, \boldsymbol{\vartheta}) &\mapsto f(\mathbf{x}, \boldsymbol{\vartheta}) \end{aligned} \tag{1}$$

be continuously differentiable with respect to both $\mathbf{x} \in S_1$ and $\boldsymbol{\vartheta} \in Q$, invertible with respect to $\mathbf{x}$, with $S_1 \subseteq \mathbb{R}^N$, $S_2 \subseteq \mathbb{R}^N$ and $Q \subseteq \mathbb{R}^M$, and let

$$f_{\boldsymbol{\vartheta}}(\mathbf{x}) = f(\mathbf{x}, \boldsymbol{\vartheta}) \tag{2}$$

where the notation $f_{\boldsymbol{\vartheta}}(\mathbf{x})$ is used whenever the function parameters are considered constants.

Let also $\mathbf{Y}$ be a continuous random variable over S_2 with pdf $P_{\mathbf{Y}}(\mathbf{y})$, and $\mathbf{X}$ be the random variable obtained applying the inverse function:

$$\mathbf{X} = f_{\boldsymbol{\vartheta}}^{-1}(\mathbf{Y}) \tag{3}$$

The pdf of $\mathbf{X}$ is given by [17] (pp.149–150):

$$P_{\mathbf{X}}(\mathbf{x}) = P_{\mathbf{Y}}(f_{\boldsymbol{\vartheta}}(\mathbf{x})) \log \left| \mathbf{J}_{\mathbf{x}}^{f_{\boldsymbol{\vartheta}}}(\mathbf{x}) \right| , \tag{4}$$

where $\mathbf{J}_{\mathbf{x}}^{f_{\boldsymbol{\vartheta}}}(\mathbf{x})$ is the Jacobian of $f_{\boldsymbol{\vartheta}}(\mathbf{x})$ w.r.t. $\mathbf{x}$, computed at $\mathbf{x}$, with elements (i, j):

$$\mathbf{J}_{\mathbf{x}}^{f_{\boldsymbol{\vartheta}}}(\mathbf{x})_{i,j} = \left. \frac{\partial f_{\boldsymbol{\vartheta},i}}{\partial x_j} \right|_{\mathbf{x}} , \tag{5}$$

and $|\cdot|$ denotes the absolute value of the determinant. Considering the parameters $\boldsymbol{\vartheta}$ as the variables to be estimated, we can rewrite $\mathbf{J}_{\mathbf{x}}^{f_{\boldsymbol{\vartheta}}}(\mathbf{x})$ as:

$$\mathbf{J}_{\mathbf{x}}^{f_{\boldsymbol{\vartheta}}}(\mathbf{x}) = \mathbf{J}_{\mathbf{x}}^{f}(\mathbf{x}, \boldsymbol{\vartheta}). \tag{6}$$

The ML estimate of the parameters $\boldsymbol{\vartheta}$ is more easily performed using as objective function the logarithm of the probability $P_{\mathbf{X}}(\mathbf{x})$, which requires the evaluation of the gradients:

$$\nabla_{\boldsymbol{\vartheta}} \left(\log P_{\mathbf{Y}} \left(f(\mathbf{x}, \boldsymbol{\vartheta}) \right) + \log \left| \mathbf{J}_{\mathbf{x}}^{f}(\mathbf{x}, \boldsymbol{\vartheta}) \right| \right) . \tag{7}$$

Since in the next section we will use a cascade of density function transformations, it is worth recalling the composition of transformation functions [18].

Let $f_{i,\boldsymbol{\vartheta}_i}(\mathbf{x}) = f_i(\mathbf{x}, \boldsymbol{\vartheta}_i)$, $i = n, \ldots, 2, 1$, be a set of n continuously differentiable and invertible functions, with $\mathrm{dom}(f_{i+1}) = \mathrm{im}(f_i)$, and let $\boldsymbol{\vartheta} = (\boldsymbol{\vartheta}_n, \ldots, \boldsymbol{\vartheta}_2, \boldsymbol{\vartheta}_1)$ be the set of the corresponding parameters. Applying k of these functions to $\mathbf{x}$ gives:

$$\begin{aligned} \mathcal{F}_k(\mathbf{x}, \boldsymbol{\vartheta}) &= \mathcal{F}_{k,\boldsymbol{\vartheta}}(\mathbf{x}) \\ &\doteq \left(f_{k,\boldsymbol{\vartheta}_k} \circ \cdots \circ f_{2,\boldsymbol{\vartheta}_2} \circ f_{1,\boldsymbol{\vartheta}_1} \right)(\mathbf{x}) \end{aligned} \tag{8}$$

Let, for convenience, set $\mathcal{F}_0(\mathbf{x}, \boldsymbol{\vartheta}) = \mathbf{x}$. Noting that the transformation function (8) can be rewritten as:

$$\mathcal{F}_k(\mathbf{x}, \boldsymbol{\vartheta}) = f_k(\mathcal{F}_{k-1}(\mathbf{x}, \boldsymbol{\vartheta}), \boldsymbol{\vartheta}_k), \tag{9}$$

and also recalling that $\mathbf{X} = \mathcal{F}_{n,\boldsymbol{\vartheta}}^{-1}(\mathbf{Y})$, the log–pdf of $\mathbf{X}$ becomes:

$$\begin{aligned} \log P_{\mathbf{X}}(\mathbf{x}) &= \log P_{\mathbf{Y}}(\mathcal{F}_n(\mathbf{x}, \boldsymbol{\vartheta})) + \log \left| \mathbf{J}_{\mathbf{x}}^{\mathcal{F}_n}(\mathbf{x}, \boldsymbol{\vartheta}) \right| \\ &= \log P_{\mathbf{Y}}(\mathcal{F}_n(\mathbf{x}, \boldsymbol{\vartheta})) + \\ &\quad \sum_{i=1}^{n} \log \left| \mathbf{J}_{\mathbf{x}}^{f_i} \left(\mathcal{F}_{i-1}(\mathbf{x}, \boldsymbol{\vartheta}), \boldsymbol{\vartheta}_i \right) \right| \end{aligned} \tag{10}$$

The gradient expressions of this objective function with respect to the parameters $\boldsymbol{\vartheta}$ can be derived by means of a forward and a backward recursion. The gradients are passed as arguments, together with the objective function, to a BFGS optimizer for obtaining the parameters that maximize the log–probability of the development set.

3. I–vector post–processing

In this section we present the building blocks, consisting of a composition of affine and non–linear functions, which allow us to estimate the pdf of the i–vectors produced by a generic extraction module. Since G–PLDA assumes a Gaussian distribution of the i–vectors, it is natural to select as pdf for $\mathbf{Y}$ the standard normal:

$$P_{\mathbf{Y}}(\mathbf{y}) = \mathcal{N}(\mathbf{0}, \mathbf{I}) . \tag{11}$$

In the following, the pdf associated to a function f will refer to the pdf of (4), where $f_{\boldsymbol{\vartheta}} = f$, and $P_{\mathbf{Y}}(\mathbf{y})$ is given by (11).

A simple and flexible non–linear function that fits our aims is the "sinh–arcsinh transformation" in [19]:

$$\overline{f}(x, \delta, \varepsilon) = \sinh(\delta \sinh^{-1}(x) + \varepsilon) , \tag{12}$$

It can be generalized for n–dimensional variables as:

$$f(\mathbf{x}, \boldsymbol{\delta}, \boldsymbol{\varepsilon}) = \begin{bmatrix} \overline{f}(x_1, \delta_1, \varepsilon_1) \\ \vdots \\ \overline{f}(x_N, \delta_n, \varepsilon_n) \end{bmatrix} , \tag{13}$$

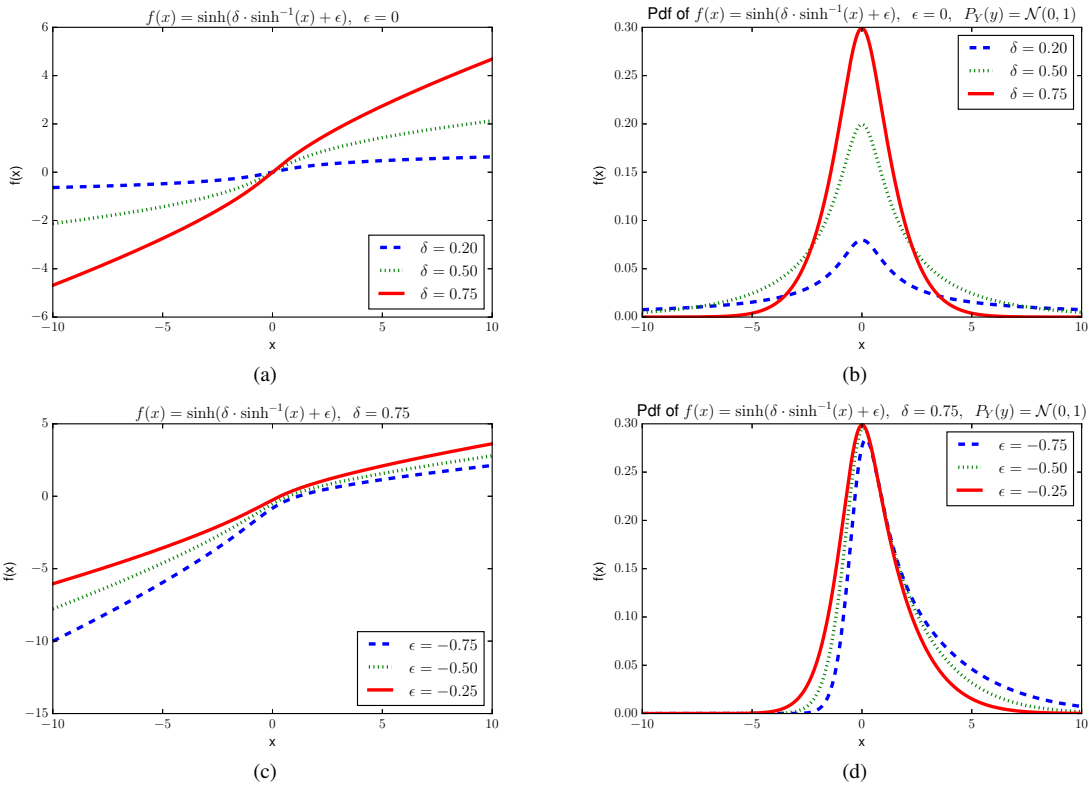


Figure 1: (a) Plot of sinh-arcsinh transformation functions, with fixed $\varepsilon = 0$, and variable value of δ. (b) Pdf of the corresponding sinh-arcsinh functions. (c) and (d) Same as (a) and (b) but with fixed $\delta = 1.0$, and variable value of ε.

where $\delta_i > 0, \quad i = 1, \ldots, n$ and ε_i control the tailweight and skewness of the distribution of each variable, respectively. We will refer in the following to this function as SAS. Figure 1a plots a family of SAS functions of a mono-dimensional variable, with fixed $\varepsilon = 0$, and variable value of δ, whereas Figure 1b plots the corresponding pdfs. Figure 1c and Figure 1d show the same plots of the previous figures, but with fixed $\delta = 1.0$, and variable value of ε. It can be noticed that by changing the two parameters of the SAS function, a wide variety of mappings can be performed, ranging from linear mapping (with $\varepsilon = 0$ and $\delta = 1.0$, which keeps a standard normal distribution, shown by the red curves), to semi–heavy–tailed symmetric or skewed distributions (see Figure 1b, and Figure 1d, respectively).

We can also observe that if:

$$\mathbf{y} = f(\mathbf{x}, \boldsymbol{\delta}, \boldsymbol{\varepsilon}) = f_{\boldsymbol{\delta},\boldsymbol{\varepsilon}}(\mathbf{x}) \,, \tag{14}$$

is the forward SAS transformation, the inverse transformation belongs to the same class, and can be easily obtained as:

$$\mathbf{x} = f^{-1}_{\boldsymbol{\delta},\boldsymbol{\varepsilon}}(\mathbf{y}) = f(\mathbf{y}, \boldsymbol{\delta}^{-1}, -\boldsymbol{\delta}^{-1} \circ \boldsymbol{\varepsilon}) \tag{15}$$

where $\boldsymbol{\delta}^{-1}$ denotes the element–wise inverse of $\boldsymbol{\delta}$ and $\circ$ the element–wise product. The inverse transformation, which can be used for sampling a distribution and generating a transformed sample, is not necessary for i–vector post–processing, because our goal is to estimate the i–vector unknown distribution, and to transform it to better fit the Gaussian assumptions by the G–PLDA model.

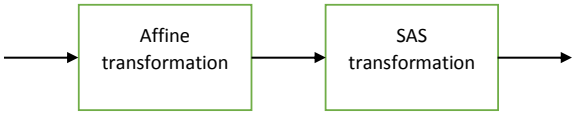


Figure 2: Basic block of transformation functions.

The second transformation that we propose to apply to i–vectors is an affine transformation defined by the function:

$$f(\mathbf{x}, \mathbf{A}, \mathbf{b}) = \mathbf{A}\mathbf{x} + \mathbf{b} \,, \tag{16}$$

where $\mathbf{A}$ is a full–rank matrix, and $\mathbf{b}$ is an offset vector.

The pdf transformation building block that we propose is the concatenation of these two functions, shown in Figure 2. We will refer to this module as an AS (Affine–SAS) block. The aim of the first affine transformation is to de–correlate the i–vector variables so that they can be independently transformed by the SAS function, and to re–scale their values toward the range that is most useful for the SAS function.

A number of AS blocks can be concatenated to form a more complex model. For example, the samples of a multi–modal distribution can be transformed into samples approximately distributed according to the standard normal distribution by estimating the parameters of a chain of AS blocks, terminated by an additional affine function.

We assessed the potential of this approach using artificial mono–dimensional data.

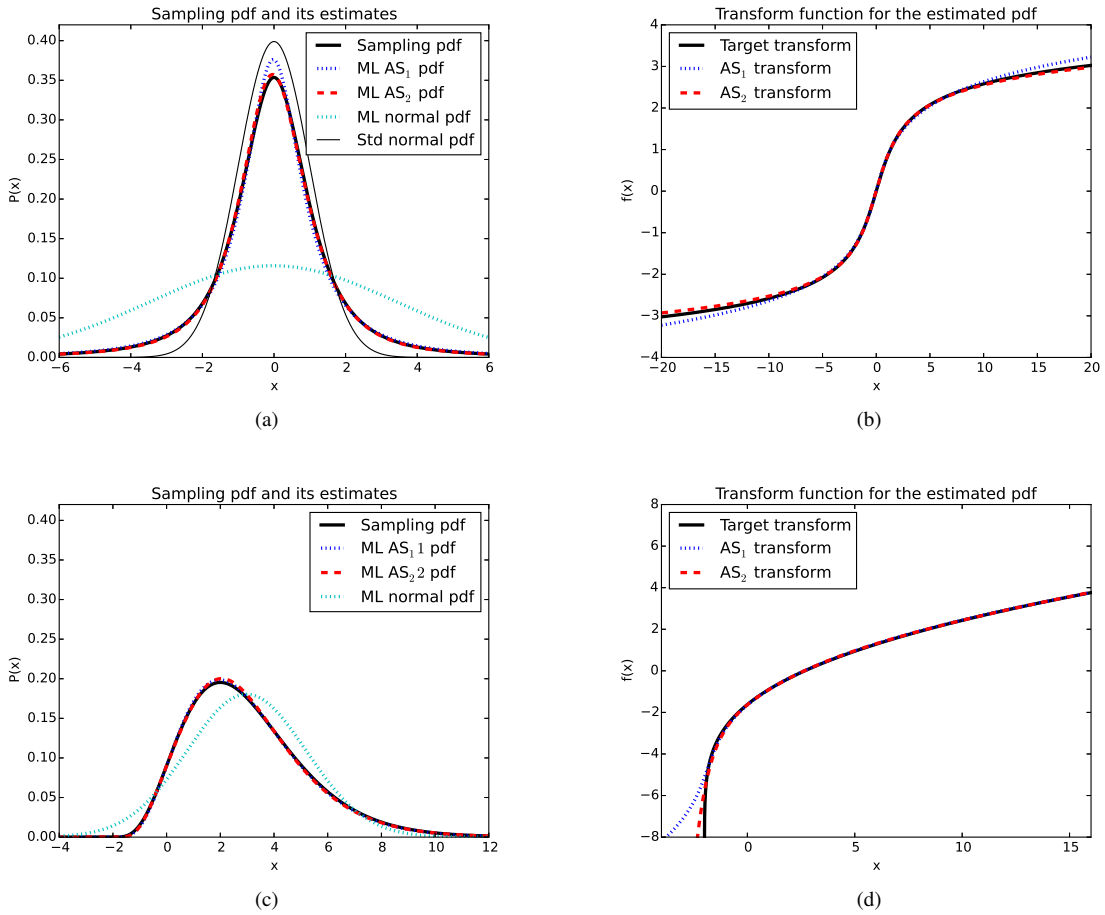


Figure 3: (a) Estimation of the pdf of a t–student distribution with two degrees of freedom by means of a one or two AS blocks (AS_1 and AS_2, respectively). (b) Corresponding transfer functions. (c) and (d) Same as (a) and (b), but for a Gamma distribution with location, scale, and shape equal to -2, 1, and 5, respectively.

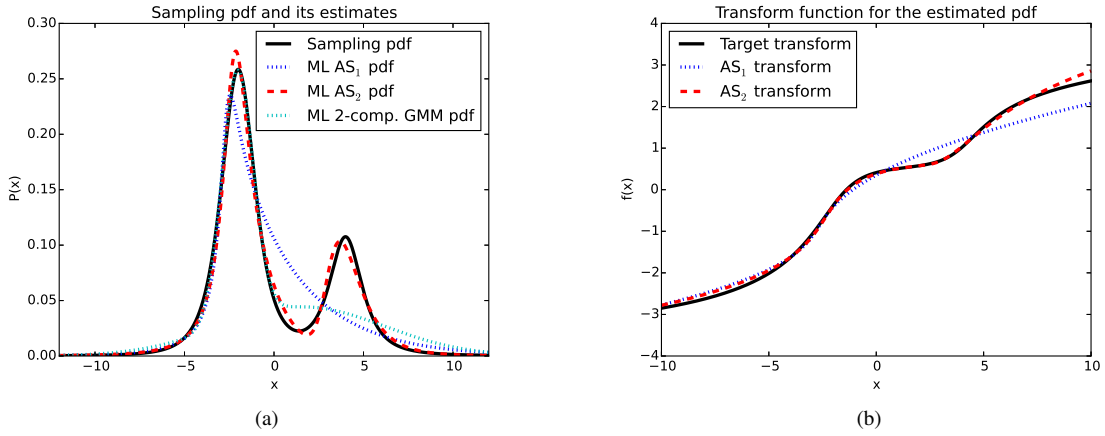
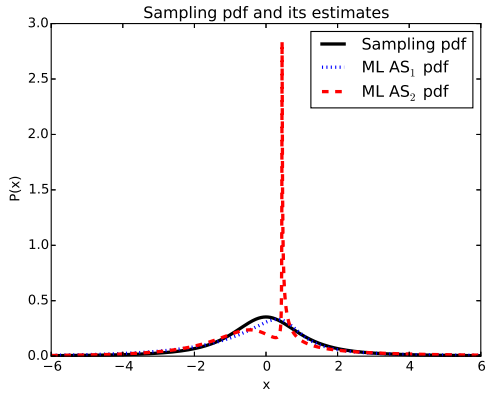
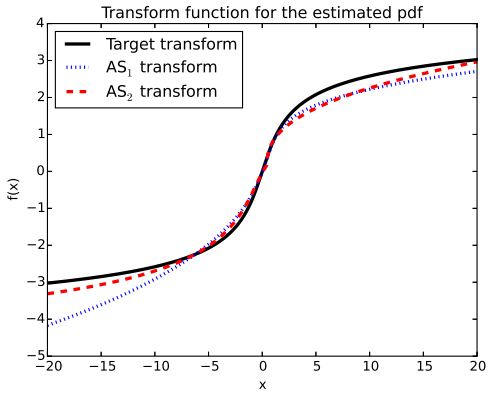


Figure 4: (a) Estimation of the pdf of a misture of two t–student distributions with parameters: weights = [0.7, 0.3], degree of freedom = [3, 2], location = [-2, 4], and scale = [1, 1] by means of a one or two AS blocks. (b) Corresponding transfer functions.



(a)



(b)

Figure 5: (a) Estimation of the pdf of a t–student distribution with two degrees of freedom by means of a one or two AS blocks. Only 50 samples of the t-student distribution available. (b) Corresponding transfer functions.

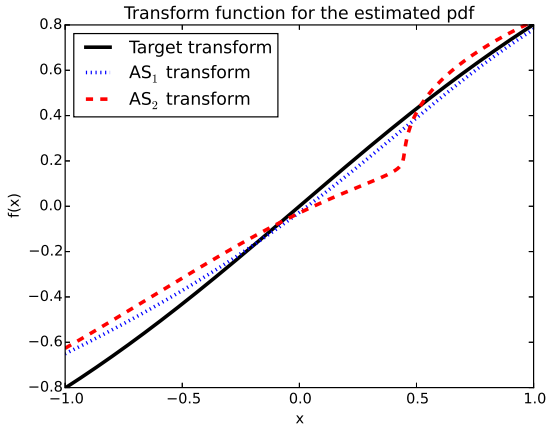


Figure 6: Zoom of a section of the transfer functions, showing a sharp derivative corresponding to the high peak in Figure 5a for the two AS block model.

We generated 10000 samples from different distributions, and estimated the parameters of a single or two AS block functions. Some results are shown in Figures 3 to 6, where label AS_1 and AS_2 refer to the single block and two AS blocks, respectively. Figure 3a has been obtained using samples generated by a t–student distribution with two degrees of freedom. It shows the original t–student distribution, which is heavy–tailed compared with a reference standard Gaussian pdf, also shown are the Gaussian obtained by ML estimation, and the distributions obtained after AS transformations. The corresponding transformation functions are plotted in Figure 3b. A similar plot for a Gamma pdf with location, scale, and shape equal to -2, 1, and 5, respectively, is shown in Figure 3c and Figure 3d. Of course, a single ML estimated Gaussian cannot fit a t–student distribution, because it tries to fit the tails of the distribution, whereas a normal distribution, transformed by an AS_2 function, provides a much better fitting of the original pdf. AS transformations do also a good job fitting the Gamma distribution even if it is worth noting that a Gamma distribution is defined in the interval $(0, \infty)$, whereas its AS approximation can also generate negative samples.

A chain of AS blocks allows also approximating a multi–modal distribution as shown in Figure 4a and Figure 4b, where the original samples are generated from a mixture of two t–student distributions with parameters: weights = [0.7, 0.3], degrees of freedom = [3, 2], location = [-2, 4], and scale = [1, 1]. In this case, even a mixture of two Gaussians does not fit well the original distribution.

Finally, the plots of Figure 5a and Figure 5b are an example of incorrect estimation, due to the scarcity of samples. The original pdf is the same t–student distribution with two degrees of freedom of Figure 3a, but the estimation is performed with 50 samples only. In this case, the AS_2 model has too much freedom (and too many parameters), thus it generates a distribution with two sharp peaks. The highest peak corresponds to the sharp derivative visible in Figure 6, which is a magnified version of a section of the transformation functions shown in figure 5b.

We will show, in Section 5, devoted to the experimental results, that this density transformation approach allows producing better i–vectors for PLDA, i.e., i–vectors that obtain better results using PLDA with respect to the standard i–vectors.

4. I–vector scaling

As stated in [14], LN allows reducing the mismatch between the development and test i-vector length distributions.

We propose a different technique to estimate scaling factors that aim at compensating this dataset mismatch. We assume that i–vectors are affected by independent scaling factors: an i–vector $\boldsymbol{\phi}_i$ is generated by a random variable whose pdf is described by the transformation:

$$\begin{aligned} \overline{\boldsymbol{\Phi}_i} &= \alpha_i^{-1} f_{\boldsymbol{\vartheta}}^{-1}(\boldsymbol{\Phi}) \\ &= (f_{\boldsymbol{\vartheta}} \circ g_{\alpha_i})^{-1}(\boldsymbol{\Phi}) \,, \end{aligned} \tag{17}$$

where $\boldsymbol{\Phi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, α_i is an i–vector dependent scaling term, and $g_{\alpha_i}(\mathbf{x}) = \alpha_i \mathbf{x}$. The terms α_i can be obtained by ML estimation similarly to the $\boldsymbol{\vartheta}$ parameters.

It is worth noting that LN can be obtained as an approximate solution of our AS transformation function, when it degenerates to a linear function, i.e., the SAS parameters are set

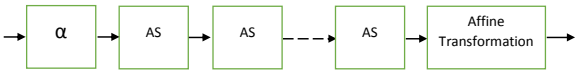


Figure 7: Chain of transformation functions including the scaling factor block.

to $\delta = 1$ and $\varepsilon = 0$, respectively, and are not re–estimated, and i–vectors have zero mean.

Let f be the linear transformation:

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}\,. \tag{18}$$

The distribution for i–vector $\boldsymbol{\phi}_i$ is given by:

$$\boldsymbol{\phi}_i \sim \mathcal{N}(\mathbf{0}, \alpha_i^{-2}\mathbf{A}^{-1}\mathbf{A}^{-T})\,. \tag{19}$$

ML optimization can be performed by alternating the estimation of the parameters of $\mathbf{A}$ and of each α_i. Starting from $\alpha_i = 1$, an optimal solution for $\mathbf{A}$ is given by:

$$\mathbf{A}^T\mathbf{A} = \boldsymbol{\Sigma}^{-1}\,, \tag{20}$$

where $\boldsymbol{\Sigma}$ denotes the i–vector covariance (assuming zero–mean i–vectors). Given $\mathbf{A}$, the ML estimate α_i^{ML} of each α_i is then:

$$\left(\alpha_i^{ML}\right)^{-1} = \sqrt{\frac{\boldsymbol{\phi}_i^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\phi}}{D}}\,, \tag{21}$$

where D is the dimension of the i–vectors. Applying the full transformation $f_{\boldsymbol{\vartheta}} \circ g_{\alpha_i}$ to an i–vector using these estimates leads to the classical length–normalized i–vector (up to a linear transformation, which is irrelevant for PLDA).

In general, the effect of the application of the function g_{α_i} can be interpreted as a length normalization tuned for the i–vector distribution described by the transformation $f_{\boldsymbol{\vartheta}}$.

In order to estimate both the parameters of the transformation $\boldsymbol{\vartheta}$ and the scaling factors α_i we adopt an iterative procedure. During training, the parameters $\boldsymbol{\vartheta}$, which are shared among all i–vectors, and the parameters α_i, which are i–vector dependent, are alternatively estimated. At testing time, we only estimate the parameters α_i of the test i–vectors. Once the parameters are estimated, we apply the transformation $f_{\boldsymbol{\vartheta}} \circ g_{\alpha_i}$ to each i–vector. The full chain of transformations is shown in Figure 7.

5. Experiments

The performance of the proposed approaches has been mostly assessed performing a set of experiments on the NIST extended core SRE 2010 female tests [20]. Other experiments, that confirm the results of the former tests, were also performed on the SRE 2012 [21].

For the SRE 2010 experiments we used cepstral features, extracted using a 25 ms Hamming window. We extract every 10 ms 19 Mel frequency cepstral coefficients together with log-energy. These 20-dimensional feature vectors are subjected to short time mean and variance normalization using a 3s sliding window. Delta and double delta coefficients are then computed using a 5-frame window giving 60-dimensional feature vectors. The i–vector extractor is based on a 2048–component full covariance gender–independent UBM, trained using NIST SRE 2004–2006 data. Gender–dependent i–vector extractors were trained using the data of NIST SRE 2004–2006, Switchboard II Phases 2 and 3, Switchboard Cellular Parts 1 and 2, Fisher English Parts 1 and 2.

The PLDA classifier was implemented according to the framework illustrated in [4]. All the experiments were performed using i–vectors with dimension $D = 400$. In these experiments the i–vector extraction post–processing includes also a preliminary Linear Discriminant Analysis (LDA), which reduced the vector dimensionality to 150. This value has been selected according to the results of previous experiments with standard i–vectors, and for reducing the complexity of the AS approach.

For the SRE 2012 experiments we used, instead, 45-dimensional feature vectors obtained by stacking 18 cepstral (c_1-c_{18}), 19 delta (Δc_0-Δc_{18}) and 8 double–delta ($\Delta\Delta c_0$-$\Delta\Delta c_7$) parameters. We trained a gender–independent i–vector extractor, based on a 1024–component diagonal covariance UBMs, estimated with data from NIST SRE 2004–2010, and additionally with the Switchboard II, Phases 2 and 3, and Switchboard Cellular, Parts 1 and 2 datasets. The i-vector dimension was again set to $d = 400$. In these experiments the i–vector extraction post–processing does not include any dimensionality reduction. Previous experiments with the baseline PLDA system have shown that LDA is not effective for this dataset, probably due to the larger number of training speakers.

The estimated model uses a single AS block and the additional final affine transformation. More complex models did not improve the performance. The BFGS optimization was terminated when the log-likelihood of the development data stopped improving.

A set of experiments were also performed on the SRE 2010 evaluation using the Spherical Nuisance normalization applied to the development and test i–vectors [16].

6. Results

Table 1 summarizes the results of the evaluated approaches on the female part of all extended core conditions in the NIST 2010 evaluation. The recognition accuracy is given in terms of percent Equal Error Rate (EER) and Minimum Detection Cost Function defined by NIST for that evaluation (minDCF10). The scores are not normalized. Last column shows the percentage of the average minDCF10 improvement with respect to the baseline G–PLDA system using length normalized i–vectors.

The performance of the G–PLDA system using i–vectors without length normalization is shown in the first row of Table 1. Excluding the EER in condition 1, all other results show a significant improvement. This suggests that the gaussianization of the i–vector pdf is effective for PLDA classification, giving on average approximately 16% improvement of minDCF10.

In the fourth and fifth rows of Table 1 it is possible compare the performance of the Spherical Nuisance normalization, after a single or three iterations (no improvements was observed using more iterations).

The last three rows of the table show the improvement obtained by iterating the estimation of the parameters of the AS model, and of the scaling factors. One can observe that our approach achieves approximately 13% average improvement with respect to G–PLDA with LN.

Finally, the row labeled "AS with LN" refers to a system that uses the AS model, but replaces the scaling factor function with the standard LN. This comparison is useful to assess the importance of using both techniques of our proposed approach. Overall, it can be noticed that the largest performance improvement

Table 1: Results for the core extended NIST SRE2010 female tests in terms of % EER and minDCF10 using different models. α–AS refers to the AS model with scaling factor.

System	Cond 1		Cond 2		Cond 3		Cond 4		Cond 5		DCF10 average
	EER	DCF10	EER	DCF10	EER	DCF10	EER	DCF10	EER	DCF10	improvement
G–PLDA	2.06	0.288	3.60	0.541	3.27	0.481	1.71	0.335	3.91	0.417	-
AS without scaling	2.15	0.221	3.36	0.462	2.96	0.414	1.61	0.290	3.19	0.391	-
G–PLDA with LN	1.81	0.255	2.83	0.476	1.95	0.367	1.21	0.295	2.19	0.347	0 %
G–PLDA with Sph iter. 1	1.81	0.254	2.60	0.458	2.04	0.379	1.15	0.303	2.08	0.351	-0,3 %
G–PLDA with Sph iter. 3	1.88	0.249	2.53	0.448	2.04	0.372	1.15	0.298	2.08	0.352	1,2 %
AS with LN	1.63	0.223	2.86	0.432	2.25	0.402	1.31	0.273	2.06	0.344	3,8 %
α–AS iter. 1	1.80	0.204	2.83	0.424	2.15	0.373	1.20	0.280	2.03	0.333	7,2 %
α–AS iter. 2	1.63	0.192	2.61	0.408	2.20	0.355	1.14	0.237	2.24	0.345	11,7 %
α–AS iter. 3	1.38	0.192	2.58	0.406	2.30	0.361	1.20	0.237	2.16	0.322	12,8 %

Table 2: $C_{primary}$ for the core extended NIST SRE 2012 tests using different models. α–AS refers to the AS model with scaling factor.

System	Cond 1 interview without added noise	Cond 2 phone call without added noise	Cond 3 interview with added noise	Cond 4 phone call with added noise	Cond 5 phone call noisy environment
G–PLDA	0.311	0.429	0.245	0.590	0.486
AS without scaling	0.337	0.446	0.275	0.615	0.497
G–PLDA with LN	0.316	0.323	0.255	0.457	0.366
α–AS iter. 1	0.268	0.313	0.236	0.474	0.357
α–AS iter. 2	0.264	0.301	0.246	0.472	0.342
α–AS iter. 3	0.261	0.299	0.240	0.470	0.342

is obtained on the microphone conditions (Cond 1, 3, and 4), suggesting that the i–vectors extracted in these conditions most benefit of the proposed transformations.

The tests on the SRE 2012 show that AS without scaling gives worse results with respect to G–PLDA without LN. We believe that this is caused by two effects. The i–vector dimensions are not reduced, thus the model has more parameters, which can lead the overfitting. Moreover, it is possible that the mismatch between the development and test i–vectors is more relevant, thus its effects could be amplified by the non–linear transformation. On conditions 1, 2, and 5, which do not include artificial added noise in test, AS with scaling confirms its effectiveness with respect to G–PLDA with LN. Since the artificial noise does not appear in the development set, for the other two conditions, our approach is probably less effective in modeling the distribution of the test i–vectors, thus it produces worse results.

7. Conclusions

We have presented a method for transforming the i–vectors so that their distribution becomes more suitable to discriminate speakers using PLDA. We employ a sequence of affine and non–linear transformations, the parameters of which are obtained by Maximum Likelihood (ML) estimation on the development set. We have also proposed a complementary technique to address the mismatch between the development and test i–vector distributions. Our approach is beneficial for G–PLDA based speaker recognition, in particular for the microphone conditions. Although LN is a fast and effective technique, we achieved a significant improvement (up to 13% relative, on average). Our i–vector processing is more complex, but its computational cost is comparable to the standard i–vector extraction, thus it does not sensibly affect the PLDA classification costs.
Future work will be devoted to the evaluation of the capability and limits of more complex models.

8. Acknowledgments

We would like to thank Niko Brummer from Agnitio for triggering our interest in the topic of density model transformations, during the Torino Speaker Recognition Summer Workshop 2015 (TOSREW 2015). We are also indebted with Oldřich Plchot from Brno University of Technology for providing the NIST SRE 2010 i–vectors.

9. References

[1] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front–end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[2] P. Kenny, "Bayesian speaker verification with heavy–tailed priors," in *Keynote presentation, Odyssey 2010, The Speaker and Language Recognition Workshop*, 2010. Available at `http://www.crim.ca/perso/patrick.kenny/kenny_Odyssey2010.pdf`.

[3] N.Brummer, "A farewell to SVM: Bayes factor speaker detection in supervector space," 2006. Available at `https://sites.google.com/site/nikobrummer/`.

[4] N. Brümmer and E. de Villiers, "The speaker partitioning problem," in *Proc. Odyssey 2010*, pp. 194–201, 2010.

[5] S. Cumani and P. Laface, "Large scale training of pairwise support vector machines for speaker recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 11, pp. 1590–1600, 2014.

[6] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted Gaussian Mixture Models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 31–44, 2000.

[7] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware Deep Neural Networks," in *Proceedings of ICASSP 2014*, pp. 1695–1699, 2014.

[8] P. Kenny, V. Gupta, T. Stafylakis, P. Ouellet, and J. Alam, "Deep Neural Networks for extracting Baum-Welch statistics for speaker recognition," in *Proceedings of Odyssey 2014*, pp. 293–298, 2014.

[9] S. Cumani, P. Laface, and F. Kulsoom, "Speaker recognition by means of acoustic and phonetically informed GMMs," in *Proceedings ofInterspeech 2015*, pp. 200–204, 2015.

[10] M. McLaren, Y. Lei, and L-Ferrer, "Advances in Deep Neural Network approaches to speaker secognition," in *Proceedings of ICASSP 2015*, pp. 4814–4818, 2015.

[11] F. Richardson, D. A. Reynolds, and N. Dehak, "A unified Deep Neural Network for speaker and language recognition," in *Proceedings ofInterspeech 2015*, pp. 1146–1150, 2015.

[12] P. Matějka, O. Glembek, O. Novotny, O. Plchot, F. Grétzl, L. Burget, and J. Černocký, "Analysis of DNN approaches to speaker identification." Submitted to ICASSP 2016.

[13] Y. Liu, Y. Qian, N. Chen, T. Fu, Y. Zhang, and K. Yu, "Deep feature for text-dependent speaker verification," *Speech Communications*, vol. 73, pp. 1–13, October 2015.

[14] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i–vector length normalization in speaker recognition systems," in *Proc. of Interspeech 2011*, pp. 249–252, 2011.

[15] P.-M. Bousquet, D.Matrouf, and J.-F. Bonastre, "Intersession compensation and scoring methods in the i-vectors space for speaker recognition," in *"Proceedings of INTERSPEECH 2011"*, pp. 485–488, 2011.

[16] P. Bousquet, A. Larcher, D.Matrouf, J.-F. Bonastre, and O. Plchot, "Variance-spectra based normalization for i-vector standard and probabilistic linear discriminant analysis," in *"Proceedings of Odyssey 2012"*, pp. 157–164, 2012.

[17] D. J. Poirier, *Intermediate statistics and econometrics: a comparative approach*. MIT Press, 1995.

[18] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015. Available at `http://jmlr.org/proceedings/papers/v37/rezende15.pdf`.

[19] M. C. Jones and A. Pewsey, "Sinh–arcsinh distributions," *Biometrika*, vol. 96, no. 4, pp. 761–780, 2009.

[20] "The NIST year 2010 speaker recognition evaluation plan." Available at `http://www.itl.nist.gov/iad/mig/tests/sre/2010/NIST_SRE10_evalplan.r6.pdf`.

[21] "The NIST year 2012 speaker recognition evaluation plan." Available at "`http://www.nist.gov/itl/iad/mig/upload/NIST_SRE12_evalplan-v17-r1.pdf`.