1

Joint estimation of PLDA and non–linear transformations of speaker vectors

Sandro Cumani and Pietro Laface

***Abstract*—The Gaussian Probabilistic Linear Discriminant Analysis (PLDA) model assumes Gaussian distributed priors for the latent variables that represent the speaker and channel factors. Assuming that each training i–vector belongs to a different speaker, as is usually done in i–vector extraction, i–vectors generated by a PLDA model can be considered independent and identically distributed with Gaussian distribution. Thus, we have recently proposed to transform the development i–vectors so that their distribution becomes more Gaussian–like. This is obtained by means of a sequence of affine and non–linear transformations whose parameters are trained by Maximum Likelihood (ML) estimation on the development set. The evaluation i–vectors are then subject to the same transformation.**

Although the i–vector "gaussianization" has shown to be effective, since the i–vectors extracted from segments of the same speaker are not independent, the original assumption is not satisfactory. In this work we show that the model can be improved by properly exploiting the information about the speaker labels, which was ignored in the previous model. In particular, a more effective PLDA model can be obtained by jointly estimating the PLDA parameters and the parameters of the non–linear transformation of the i–vectors. In other words, while the goal of the previous approach was to "gaussianize" the training i–vectors distribution, the objective of this work is to embed the estimation of the non–linear i–vector transformation in the PLDA model estimation. We will thus refer to this model as the non–linear PLDA model (NL–PLDA).

We show that this new approach provides significant gain with respect to PLDA, and a small, yet consistent, improvement with respect to our former i–vector "gaussianization" approach, without further additional costs.

***Index Terms*—Speaker recognition, i–vector, PLDA, non–linear density transformation.**

I. INTRODUCTION

Most speaker recognition systems are based on Gaussian Mixture Models (GMMs) [1], where a speech segment is represented by a compact "identity vector" (i–vector for short) [2], extracted by means of Factor Analysis. The main advantage of this representation is that the problem of intersession variability is deferred to a second stage, dealing with low-dimensional vectors rather than with the high-dimensional space of the GMM means. In particular, systems based on i–vectors, extracted by means of the hybrid Deep Neural Network (DNN) and GMM approach, [3]–[6], and on Probabilistic Linear Discriminant Analysis (PLDA) [7]–[9], or discriminative classifiers [10]–[13], represent the current state–of–the–art in text–independent speaker recognition.

The authors are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, 10143 Torino, Italy (e-mail: sandro.cumani@polito.it, pietro.laface@polito.it).
Computational resources for this work were provided by HPC@POLITO (http://www.hpc.polito.it)

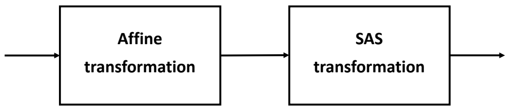

Fig. 1: Block diagram of the transformation functions.

A number of alternative approaches based on Restricted Boltzmann Machines or DNN, alternative to the hybrid DNN/GMM technique, have been proposed in the last years for speaker–dependent and speaker–independent recognition. Indeed, the success of DNNs for speech recognition [14] has fostered the study of new DNN architectures for speaker recognition with the aim of replacing totally or in part the traditional GMM approach [15]–[20]. In particular, based on the success of the DNN techniques in face recognition [21], the so called end–to–end speaker verification is actively being explored with interesting results [22], [23].

Although we believe that the end–to–end architectures pave the way for the next generation of state–of–the–art systems, we have recently shown that there is still place for improvement of the current state–of–the–art systems based on generative models [24]–[26]. This paper is a further contribution along this line of research, aiming at estimating more effective PLDA models.

PLDA models the underlying speaker and channel variability in the i–vector space using a probabilistic framework. From the PLDA distributions it is possible to evaluate the likelihood ratio between the "same speaker" hypothesis and "different speaker" hypothesis for a set of i–vectors. I–vector extraction is usually performed ignoring the model that will be used for classification, but it is reasonable to expect better performance if the features provided to a classifier fulfill its assumptions. In particular, the simplified Gaussian PLDA model 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 represented as the sum of three factors, namely the i–vector global mean $\mathbf{m}$, a speaker factor $\mathbf{y}$, and a factor ϵ that represents the inter–session and the residual noise, both Gaussian distributed, as:

$$\boldsymbol{\phi} = \mathbf{m} + \mathbf{U}\mathbf{y} + \epsilon \ , \tag{1}$$

where matrix $\mathbf{U}$ can constrain the speaker factors to be of lower dimension than the i–vectors space.

PLDA estimates the model parameters that maximize the likelihood of the observed i–vectors, assuming that the i–vectors of a given speaker share the same speaker factor, i.e., the same value for latent variable $\mathbf{y}$ [7].

Assuming that each training i–vector belongs to a different

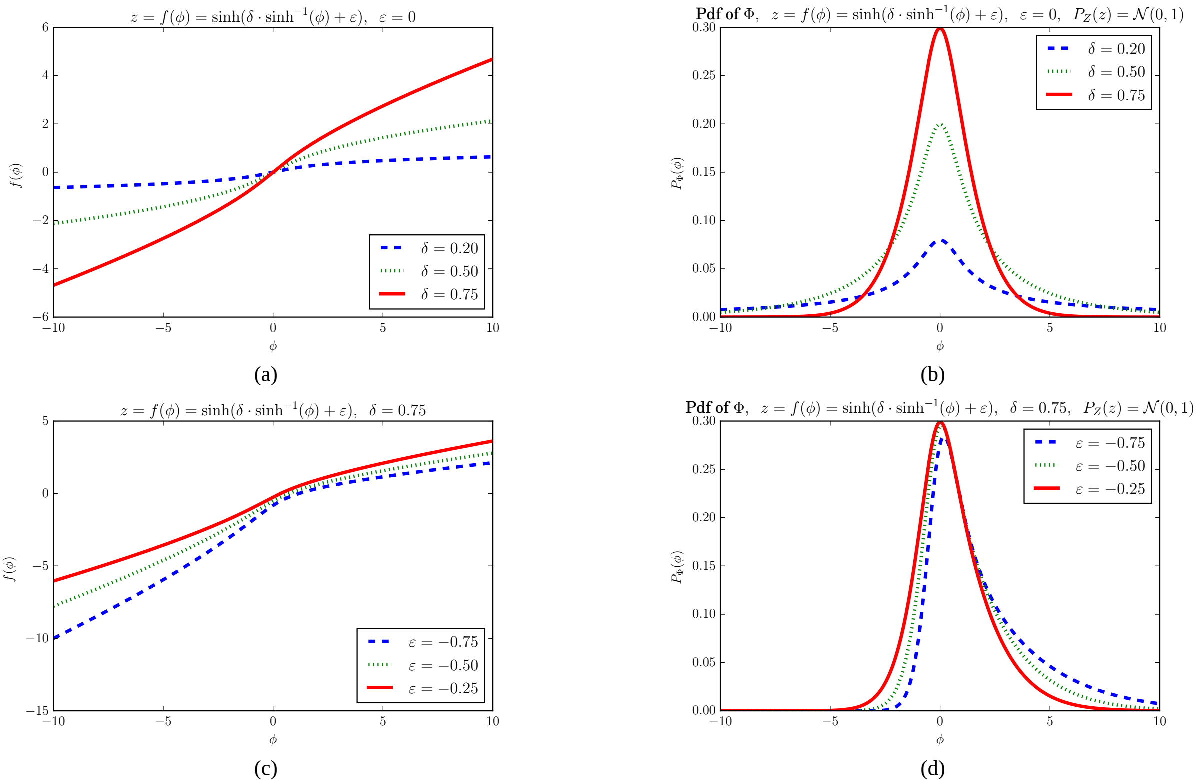

Fig. 2: (a) Plot of sinh-arcsinh transformation functions, with $\varepsilon = 0$, and variable δ. (b) Pdf of the corresponding transformed random variables. (c) and (d) Same as (a) and (b) but with $\delta = 0.75$, and variable ε.

speaker, i–vectors generated by a PLDA model can be considered independent and identically distributed with Gaussian distribution. Although this assumption is not too realistic, it is consistent with the i–vector extraction processing, where all utterances are considered independent. Thus, we have recently proposed to transform the development i–vectors so that their distribution becomes more Gaussian–like [24], [25]. This transformation is obtained by means of a sequence of affine and non–linear transformations whose parameters are trained by Maximum Likelihood (ML) estimation on the development set. We will refer to this transformation in the following as the AS (Affine–SAS) transformation because it is obtained by means of a cascade of affine and non–linear sinh-arcsinh (SAS) functions, as illustrated in Fig. 1, and detailed in Section II. We will also refer to the PLDA classifier using the AS transformed i–vectors as the AS–PLDA system.

The assumption that i–vectors are independent is not, however, satisfactory because the training set normally includes several segments for each speaker, which are of course not independent. Aiming at obtaining a more effective PLDA classifier, we present a new approach that tackles this problem by properly exploiting the information about the speaker labels, which was ignored in the previous model. We propose to embed the estimation of the non-linear i–vector transformation in the PLDA model estimation, i.e., to jointly estimate the PLDA parameters and the parameters of the non–linear transformation of the i–vectors. In other words, while the goal of the previous approach was to “gaussianize” the training i–vectors distribution, the objective of this work is to embed the estimation of the non–linear i–vector transformation in the PLDA model estimation. We will thus refer to this model as the non–linear PLDA model (NL–PLDA).
The results of this approach will be compared with the standard PLDA, and with the AS–PLDA approach on the NIST 2012 evaluation trials, using different systems and classifiers, and a recently proposed compact representation of a speech segment, similar to i–vector, which we have called e–vector [26]. We rely on e–vectors because we have shown in [26] that simply replacing the i–vectors with e–vectors, we have obtained approximately 10% average improvement of the C_{primary} cost function on the NIST 2012 evaluation trials, using different systems and classifiers.

The paper is organized as follows: In Section II we briefly recall the non–linear transformations proposed in [24], [25]. Section III introduces our new generative non–linear PLDA model, which allows us to embed the estimation of the non-linear i–vector transformation in the PLDA model estimation. NL–PLDA model training and scoring are illustrated in Sections IV, and V, respectively. In Section VI we detail the scaling–factor normalization technique, which is our alternative to i–vector length normalization for the non–linear models, and we illustrate NL–PLDA scoring with scaling–factor normalization. The e–vector representation, its relation with i–vectors, and its estimation procedure are briefly recalled in Section VII. Section VIII is devoted to the experimental settings and results, and conclusions are drawn in Section IX.

II. DENSITY FUNCTION TRANSFORMATIONS

The non–linear i–vector transformation model proposed in [24], [25] assumes that i–vectors are independently sampled from a standard normal distribution, and independently transformed by means of an invertible non–linear function f^{-1}:

$$\mathbf{\Phi} = f^{-1}(\mathbf{Z}) \ , \tag{2}$$

where $\mathbf{\Phi}$ is a random variable that generates i–vectors, $\mathbf{Z}$ is a standard normal random variable, and $\boldsymbol{\phi} \sim \mathbf{\Phi}$ is a sampled i–vector [1]. By applying function f to i–vectors $\boldsymbol{\phi}$, they are transformed into samples that follow a standard normal distribution. The transformed i–vectors, $f(\boldsymbol{\phi})$, are then used as features for training a PLDA classifier.

This model is approximate, because it assumes that each training i–vector belongs to a different speaker, as is usually done in i–vector extraction, nevertheless since Gaussian PLDA assumes that independently sampled i–vectors, belonging to different speakers, follow a Gaussian distribution, transforming i–vectors by means of a function f estimated on a development set, allows better fitting the PLDA assumptions.

In particular, to fit a Gaussian distribution, in [24], [25] we make use of the affine transformation defined as:

$$l(\boldsymbol{\phi}, \mathbf{A}, \mathbf{b}) = \mathbf{A}\boldsymbol{\phi} + \mathbf{b} \ , \tag{3}$$

where $\mathbf{A}$ is a full–rank matrix, and $\mathbf{b}$ is an offset vector, for the linear part of the i–vector transformation. The non–linear transformation is based on the sinh–arcsinh function [27], [28]:

$$\overline{f}(\phi, \delta, \varepsilon) = \sinh(\delta \sinh^{-1}(\phi) + \varepsilon) \ , \tag{4}$$

which can be generalized for N–dimensional variables as:

$$f(\boldsymbol{\phi}, \boldsymbol{\delta}, \boldsymbol{\varepsilon}) = \begin{bmatrix} \overline{f}(\phi_1, \delta_1, \varepsilon_1) \\ \vdots \\ \overline{f}(\phi_N, \delta_N, \varepsilon_N) \end{bmatrix} \ , \tag{5}$$

where $\delta_i > 0$ controls the tailweight of the distribution, and ε_i affects the skewness of each variable. This function has been selected because it is invertible, and flexible in mapping a distribution to another distribution. Fig. 2a plots a family of SAS functions of a mono–dimensional variable, with fixed $\varepsilon = 0$, and variable value of δ, whereas Fig. 2b shows how a standard normal distribution is transformed by applying the corresponding SAS function. Fig. 2c and Fig. 2d show the same plots of the previous figures, but with fixed $\delta = 0.75$, and variable value of ε. By changing the value of 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 would keep the original standard normal distribution), to semi–heavy–tailed symmetric or skewed distributions (see Fig. 2b, and Fig. 2d, respectively).

The aim of the 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 towards the most suitable range for the SAS function. In [24] we have shown that a number of AS modules can be concatenated, leading to more accurate transformation functions.

Since both the affine and the proposed transformations are invertible it is possible to fit, for example, the samples of a multi–modal distribution by transforming the samples of the standard normal distribution.

Looking at the transformation the other way around, applying the inverse function, the samples of the original distribution can be transformed to match a standard normal distribution.

In [25] PLDA classification based on this “gaussianized” i–vectors (AS–PLDA) has been successfully tested on the NIST SRE-2010 and SRE-2012 evaluation datasets showing a relative improvement between 7% and 14% of their Detection Cost Function with respect to the use of standard i-vectors. Crucial to the success of this approach was the reduction of the mismatch between the development and evaluation i–vector length distributions by means of an i–vector dependent scaling factor, which we detail in Section VI.

III. NON–LINEAR PLDA MODEL

We here detail our approach that embeds the estimation of the non-linear i–vector transformation in the PLDA model estimation. Since we jointly estimate both the PLDA parameters, and the parameters of the non–linear transformation of the i-vectors, also for the latter we intrinsically account for the speaker information associated to each i–vector, an information that was ignored in the approach proposed in [24], [25].

A. Model description

The generative NL–PLDA model that we propose in this work is given by:

$$\begin{aligned} \mathbf{z} &= \mathbf{U}\mathbf{y} + \boldsymbol{\epsilon} \\ \boldsymbol{\phi} &= f^{-1}(\mathbf{z}) \ , \end{aligned} \tag{6}$$

where $\mathbf{y}$, $\boldsymbol{\epsilon}$ and $\mathbf{U}$ have been defined in (1). We assume that the speaker factor $\mathbf{y}$ has a standard normal prior, and that the residual term $\boldsymbol{\epsilon}$ is sampled from $\mathcal{N}(\mathbf{0}, \mathbf{\Lambda}^{-1})$, where $\mathbf{\Lambda}$ is the within–class precision matrix. Without loss of generality, we also dropped the i–vector mean $\mathbf{m}$ to simplify the equations appearing in the following.

It can be noticed that, if f is the identity transformation $f(\mathbf{a}) = \mathbf{a}$, the NL–PLDA model of (6) corresponds to the standard PLDA model with centered i–vectors.

Given model (6), we can define the following random variables, and related probability distributions:

$$\mathbf{Y} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{7}$$

$$\mathbf{Z}|(\mathbf{Y} = \mathbf{y}) \sim \mathcal{N}(\mathbf{U}\mathbf{y}, \mathbf{\Lambda}^{-1}) \tag{8}$$

$$\mathbf{\Phi}|(\mathbf{Y} = \mathbf{y}) = f^{-1}(\mathbf{Z}|\mathbf{Y} = \mathbf{y}) \tag{9}$$

$$P_{\mathbf{Y}}(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I}) \tag{10}$$

$$P_{\mathbf{Z}|\mathbf{Y}}(\mathbf{z}|\mathbf{y}) = \mathcal{N}(\mathbf{z}; \mathbf{U}\mathbf{y}, \mathbf{\Lambda}^{-1}) \tag{11}$$

$$\begin{aligned} P_{\mathbf{\Phi}|\mathbf{Y}}(\boldsymbol{\phi}|\mathbf{y}) &= P_{\mathbf{Z}|\mathbf{Y}}(f(\boldsymbol{\phi})|\mathbf{y}) \, |\mathbf{D}_{\boldsymbol{\phi}} f(\boldsymbol{\phi})| \\ &= \mathcal{N}(f(\boldsymbol{\phi}); \mathbf{U}\mathbf{y}, \mathbf{\Lambda}^{-1}) \, |\mathbf{D}_{\boldsymbol{\phi}} f(\boldsymbol{\phi})| \ , \end{aligned} \tag{12}$$

[1]In [25] $\mathbf{\Phi}$ and $\mathbf{Z}$ were denoted as $\mathbf{X}$ and $\mathbf{Y}$, respectively. Since $\mathbf{x}$ and $\mathbf{y}$ are traditionally used for the channel and speaker factors of PLDA, and we propose here the joint estimation of PLDA and transformation parameters, we decided to change the notation for the i-vectors and transformed i-vectors

where $|\mathbf{D}_{\boldsymbol\phi} f(\boldsymbol\phi)|$ is the absolute value of the determinant of the Jacobian of $f(\boldsymbol\phi)$. The probability distribution of equation (12) is obtained by exploiting the change-of-variable technique, which allows computing the pdf of a random variable function [29] (pp.149–150).

Given a set of k i–vectors belonging to the *same speaker* $\{\phi_1, \ldots, \phi_k\}$, and applying (7) and (11), the joint log–likelihood of the i–vectors and speaker factor $\mathbf{y}$ is given by:

$$\begin{aligned}
\log P(\boldsymbol\phi_1, \ldots \boldsymbol\phi_k, \mathbf{y}) &= \sum_{i=1}^{k} \left[\log P_{\boldsymbol\Phi|\mathbf{Y}}(\boldsymbol\phi_i|\mathbf{y})\right] + \log P_{\mathbf{Y}}(\mathbf{y}) \\
&= \sum_{i=1}^{k} \left[\log P_{\mathbf{Z}|\mathbf{Y}}(f(\boldsymbol\phi_i)|\mathbf{y}) + \log\left|\mathbf{D}_{\boldsymbol\phi_i} f(\boldsymbol\phi_i)\right|\right] + \log P_{\mathbf{Y}}(\mathbf{y}) \\
&= \sum_{i=1}^{k} \left[\frac{1}{2}\Big(\log|\boldsymbol\Lambda| - (f(\boldsymbol\phi_i) - \mathbf{U}\mathbf{y})^T \boldsymbol\Lambda \left(f(\boldsymbol\phi_i) - \mathbf{U}\mathbf{y}\right)\Big)\right. \\
&\qquad \left. + \log\left|\mathbf{D}_{\boldsymbol\phi_i} f(\boldsymbol\phi_i)\right|\right] - \frac{1}{2}\mathbf{y}^T\mathbf{y} + c \ , \qquad (13)
\end{aligned}$$

where all constant terms have been collected in c.

Since (13) is quadratic in $\mathbf{y}$, the posterior for $\mathbf{y}$ is Gaussian:

$$\mathbf{Y}|\left(\boldsymbol\Phi_1 = \boldsymbol\phi_1, \ldots, \boldsymbol\Phi_k = \boldsymbol\phi_k\right) \sim \mathcal{N}(\boldsymbol\mu_{\mathbf{y}}, \boldsymbol\Lambda_{\mathbf{y}}^{-1}) \ , \qquad (14)$$

with posterior precision matrix $\boldsymbol\Lambda_{\mathbf{y}}$ and mean $\boldsymbol\mu_{\mathbf{y}}$. The values of these parameters can be derived by expanding (13) and identifying the contributions of the second and first order terms, respectively, as:

$$\begin{aligned}
\boldsymbol\Lambda_{\mathbf{y}} &= \mathbf{I} + k\mathbf{U}^T\boldsymbol\Lambda\mathbf{U} \\
\boldsymbol\mu_{\mathbf{y}} &= \boldsymbol\Lambda_{\mathbf{y}}^{-1}\mathbf{U}^T\boldsymbol\Lambda \sum_{i=1}^{k} f(\boldsymbol\phi_i) \ . \qquad (15)
\end{aligned}$$

It is worth noting that (15) is very similar to the expression for the posterior of the speaker variable in standard PLDA [7]. The only difference is that the posterior mean in the NL–PLDA model is computed using the transformed i–vectors $f(\boldsymbol\phi_i)$ rather than the original i–vectors $\boldsymbol\phi_i$.

Although different, possibly more accurate, flavors of non–linear PLDA generative model can be devised, the proposed model has the substantial advantage that the conditional likelihood of the i–vectors, given a value for the speaker hidden variable $\mathbf{y}$, is conjugate to the speaker prior distribution. This allows computing posteriors, which are Gaussian distributed, in closed form, without resorting to expensive Variational Bayesian methods. Given the transformed i–vectors, scoring with the NL–PLDA model has the same computational complexity of standard PLDA.

IV. NL–PLDA TRAINING

Since we can easily compute closed form expressions for the posterior of $\mathbf{y}$, model training can be performed by means of Expectation Maximization (EM) iterations. However, since closed form solutions for the update of the parameters are not available, we have to rely on numerical optimization in the maximization steps.

In the following we will denote the set of the $k(s)$ i–vectors belonging to speaker s as $M_s = \boldsymbol\phi_{1,s}, \ldots, \boldsymbol\phi_{k(s),s}$. We will also simplify the notation by dropping the explicit distinction between random variables and samples.

The joint log–likelihood of i–vectors and speaker variables is:

$$\sum_{s=1}^{S} \log P(M_s, \mathbf{y}_s|\boldsymbol\theta) \ , \qquad (16)$$

where S is the number of speakers in the development set, and $\boldsymbol\theta$ is the set of parameters that must be optimized. The EM auxiliary function is given by the expectation of (16) with respect to $\mathbf{y}_s$:

$$Q\left(\boldsymbol\theta, \widetilde{\boldsymbol\theta}\right) = \sum_{s=1}^{S} \mathbb{E}_{\mathbf{y}_s|M_s,\widetilde{\boldsymbol\theta}} \log P(M_s, \mathbf{y}_s|\boldsymbol\theta) \ , \qquad (17)$$

where $\boldsymbol\theta$ is the new estimate of the model parameters, $\widetilde{\boldsymbol\theta}$ denotes the model parameters that have been used to compute the expectations in the E–step, and the posteriors $\mathbf{y}_s$ are computed using (15).

Substituting (13) in (17), and ignoring the terms that are irrelevant for the optimization, we obtain the objective function:

$$\begin{aligned}
Q_1(\boldsymbol\theta, \widetilde{\boldsymbol\theta}) = \sum_{s} \mathbb{E}_{\mathbf{y}_s|M_s,\widetilde{\boldsymbol\theta}} &\left[\sum_{i=1}^{k(s)} \Big(\frac{1}{2}\log|\boldsymbol\Lambda|\right. \\
&- \frac{1}{2}\big(f(\boldsymbol\phi_{i,s}, \boldsymbol\vartheta) - \mathbf{U}\mathbf{y}_s\big)^T \boldsymbol\Lambda \big(f(\boldsymbol\phi_{i,s}, \boldsymbol\vartheta) - \mathbf{U}\mathbf{y}_s\big) \\
&\left. + \log\left|\mathbf{D}_{\boldsymbol\phi_{i,s}} f(\boldsymbol\phi_{i,s}, \boldsymbol\vartheta)\right|\Big)\right] \qquad (18)
\end{aligned}$$

where we have exposed the set of parameters $\boldsymbol\vartheta$ of function f, which complete the set of model parameters $\boldsymbol\theta = [\boldsymbol\vartheta, \mathbf{U}, \boldsymbol\Lambda]$.

Before proceeding with the steps that lead us to the estimation of the model parameters, we show that $\boldsymbol\Lambda$ can be eliminated from the parameter set. Indeed, as long as f is represented as a composition of n functions $f_n(f_{n-1}(\cdot f_1(\boldsymbol\phi)))$, and f_n is affine, the model is over–parametrized.
In particular, using the function composition:

$$f(\boldsymbol\phi) = g(h(\boldsymbol\phi)) \ , \qquad (19)$$

where h is a (non–linear) transformation, and g is the affine transformation:

$$g(\tilde{\boldsymbol\phi}) = \mathbf{A}\tilde{\boldsymbol\phi} + \mathbf{b} \ , \qquad (20)$$

with $\mathbf{A}$ an invertible matrix, we can rewrite the EM auxiliary function (18) as:

$$\begin{aligned}
Q_1(\boldsymbol\theta, \widetilde{\boldsymbol\theta}) = \sum_{s} \mathbb{E}_{\mathbf{y}_s|M_s,\widetilde{\boldsymbol\theta}} &\left[\sum_{i=1}^{k(s)}\right. \\
&\Big(-\frac{1}{2}\big(\mathbf{A}h(\boldsymbol\phi_{i,s}, \boldsymbol\vartheta) + \mathbf{b} - \mathbf{U}\mathbf{y}_s\big)^T \cdot \\
&\boldsymbol\Lambda \cdot \big(\mathbf{A}h(\boldsymbol\phi_{i,s}, \boldsymbol\vartheta) + \mathbf{b} - \mathbf{U}\mathbf{y}_s\big) \\
&\left. + \frac{1}{2}\log|\boldsymbol\Lambda| + \log\left|\mathbf{D}_{\boldsymbol\phi_{i,s}}\left(\mathbf{A}h(\boldsymbol\phi_{i,s}, \boldsymbol\vartheta) + \mathbf{b}\right)\right|\Big)\right] \ . \qquad (21)
\end{aligned}$$

Considering that the term $\mathbf{b}$ is not a function of $\boldsymbol{\phi}_{i,s}$, $\log\left|\mathbf{D}_{\boldsymbol{\phi}_{i,s}}\left(\mathbf{A}h(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})+\mathbf{b}\right)\right|$ can be rewritten as:

$$\begin{aligned}\log\left|\mathbf{D}_{\boldsymbol{\phi}_{i,s}}\left(\mathbf{A}h(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})+\mathbf{b}\right)\right| &=\\ \log|\mathbf{A}|+\log\left|\mathbf{D}_{\boldsymbol{\phi}_{i,s}}h(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})\right| &. \end{aligned} \tag{22}$$

Since we can also write, for convenience:

$$\log|\mathbf{A}|=\frac{1}{2}\log(\left|\mathbf{A}\mathbf{A}^T\right|)=\frac{1}{2}\left(\log|\mathbf{A}|+\log\left|\mathbf{A}^T\right|\right)\ , \tag{23}$$

we can collect the last two terms in (21) obtaining:

$$\begin{aligned}Q_1(\boldsymbol{\theta},\widetilde{\boldsymbol{\theta}}) =& \sum_s \mathbb{E}_{\mathbf{y}_s|M_s,\widetilde{\boldsymbol{\theta}}}\left[\sum_{i=1}^{k(s)}\right.\\ &\left(-\frac{1}{2}\big(h(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})+\mathbf{A}^{-1}\mathbf{b}-\mathbf{A}^{-1}\mathbf{U}\mathbf{y}_s\big)^T\mathbf{A}^T\cdot\right.\\ &\boldsymbol{\Lambda}\cdot\mathbf{A}\left(h(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})+\mathbf{A}^{-1}\mathbf{b}-\mathbf{A}^{-1}\mathbf{U}\mathbf{y}_s\right)\\ &\left.\left.+\frac{1}{2}\log\left|\mathbf{A}^T\boldsymbol{\Lambda}\mathbf{A}\right|+\log\left|\mathbf{D}_{\boldsymbol{\phi}_{i,s}}h(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})\right|\right)\right]\ .\end{aligned} \tag{24}$$

Finally, defining:

$$\boldsymbol{\Lambda}_1=\mathbf{A}^T\boldsymbol{\Lambda}\mathbf{A} \tag{25}$$
$$\mathbf{m}_1=-\mathbf{A}^{-1}\mathbf{b} \tag{26}$$
$$\mathbf{U}_1=\mathbf{A}^{-1}\mathbf{U}\ , \tag{27}$$

we get:

$$\begin{aligned}Q_1(\boldsymbol{\theta},\widetilde{\boldsymbol{\theta}}) =& \sum_s \mathbb{E}_{\mathbf{y}_s|M_s,\widetilde{\boldsymbol{\theta}}}\left[\sum_{i=1}^{k(s)}\right.\\ &\left(-\frac{1}{2}\big(h(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})-\mathbf{m}_1-\mathbf{U}_1\mathbf{y}_s\big)^T\boldsymbol{\Lambda}_1\right.\\ &\big(h(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})-\mathbf{m}_1-\mathbf{U}_1\mathbf{y}_s\big)\\ &\left.\left.+\frac{1}{2}\log|\boldsymbol{\Lambda}_1|+\log\left|\mathbf{D}_{\boldsymbol{\phi}_{i,s}}h(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})\right|\right)\right]\ .\end{aligned} \tag{28}$$

which is equivalent, except for the mean term $\mathbf{m}_1$, to (18). Therefore, the parameters of the final affine function are able to embed the global model mean $\mathbf{m}$, which was not considered in the model for the sake of notation simplicity. Another important effect of the final affine transformation is that we can set the precision matrix $\boldsymbol{\Lambda}$ to a constant value because $\boldsymbol{\Lambda}_1$ can be obtained by estimating matrix $\mathbf{A}$, and using equation (25).

Matrix $\boldsymbol{\Lambda}$ could be set to the identity matrix, but it is more reasonable to set it to the matrix $\boldsymbol{\Lambda}$ estimated by standard PLDA. Since $\boldsymbol{\Lambda}$ is kept constant, the set of parameters to be estimated reduces to $\boldsymbol{\theta}=[\boldsymbol{\vartheta},\mathbf{U}]$, and the EM auxiliary function (18) can be simplified as:

$$\begin{aligned}Q_2\left(\boldsymbol{\theta},\widetilde{\boldsymbol{\theta}}\right) =& \sum_s \mathbb{E}_{\mathbf{y}_s|M_s,\widetilde{\boldsymbol{\theta}}}\left[\sum_{i=1}^{k(s)}\right.\\ &\left(-\frac{1}{2}\big(f(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})-\mathbf{U}\mathbf{y}_s\big)^T\boldsymbol{\Lambda}\big(f(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})-\mathbf{U}\mathbf{y}_s\big)\right.\\ &\left.\left.+\log\left|\mathbf{D}_{\boldsymbol{\phi}_{i,s}}f(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})\right|\right)\right]\ .\end{aligned} \tag{29}$$

Moving the expectation operator inside the summation operation, and omitting its subscripts for the sake of simplicity, we get:

$$\begin{aligned}Q_2\left(\boldsymbol{\theta},\widetilde{\boldsymbol{\theta}}\right) =& \sum_s \left[\sum_{i=1}^{k(s)}\right.\\ &\left(-\frac{1}{2}\big(f(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})-\mathbf{U}\mathbb{E}[\mathbf{y}_s]\big)^T\boldsymbol{\Lambda}\big(f(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})-\mathbf{U}\mathbb{E}[\mathbf{y}_s]\big)\right.\\ &-\frac{1}{2}\operatorname{Tr}\left(\mathbf{U}^T\boldsymbol{\Lambda}\mathbf{U}\operatorname{Cov}\left(\mathbf{y}_s,\mathbf{y}_s\right)\right)\\ &\left.\left.+\log\left|\mathbf{D}_{\boldsymbol{\phi}_{i,s}}f(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})\right|\right)\right]\ .\end{aligned} \tag{30}$$

Equation (30) has been obtained by using the property of the trace of a product, which is invariant under cyclical permutations of the product factors,

$$\mathbf{x}^T\mathbf{A}\mathbf{x}=\operatorname{Tr}(\mathbf{x}^T\mathbf{A}\mathbf{x})=\operatorname{Tr}(\mathbf{A}\mathbf{x}\mathbf{x}^T)\ , \tag{31}$$

furthermore, we have also used the property that expectation and trace are linear operators, which can be commuted,

$$\mathbb{E}[\mathbf{x}^T\mathbf{A}\mathbf{x}]=\operatorname{Tr}(\mathbf{A}\mathbb{E}[\mathbf{x}\mathbf{x}^T])\ , \tag{32}$$

and the properties of the covariance matrix of random vectors:

$$\mathbb{E}[\mathbf{x}\mathbf{x}^T]=\mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{x}^T]+\ \operatorname{Cov}(\mathbf{x},\mathbf{x}). \tag{33}$$

Expanding (30) we get:

$$\begin{aligned}Q_2\left(\boldsymbol{\theta},\widetilde{\boldsymbol{\theta}}\right) = -\frac{1}{2}\sum_s\sum_{i=1}^{k(s)}\Big[&\operatorname{Tr}\left(\boldsymbol{\Lambda}f(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})f(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})^T\right)\\ &-2\operatorname{Tr}\left(\boldsymbol{\Lambda}\mathbf{U}\mathbb{E}\left[\mathbf{y}_s\right]f(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})^T\right)\\ &+\operatorname{Tr}\left(\boldsymbol{\Lambda}\mathbf{U}\mathbb{E}[\mathbf{y}_s\mathbf{y}_s^T]\mathbf{U}^T\right)\\ &+\log\left|\mathbf{D}_{\boldsymbol{\phi}_{i,s}}f(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})\right|\Big]\ ,\end{aligned} \tag{34}$$

where the expectations are taken with respect to $\mathbf{y}_s|M_s,\widetilde{\boldsymbol{\theta}}$.

Recalling equations (101), and (118) of [30]:

$$\nabla_{\mathbf{U}}\operatorname{Tr}\left[\boldsymbol{\Lambda}\mathbf{U}\mathbb{E}\left[\mathbf{y}_s\right]f(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})^T\right]=\boldsymbol{\Lambda}^Tf(\boldsymbol{\phi}_{i,s},\boldsymbol{\vartheta})\mathbb{E}\left[\mathbf{y}_s^T\right]$$
$$\nabla_{\mathbf{U}}\operatorname{Tr}\left[\boldsymbol{\Lambda}\mathbf{U}\mathbb{E}[\mathbf{y}_s\mathbf{y}_s^T]\mathbf{U}^T\right]=\boldsymbol{\Lambda}^T\mathbf{U}\mathbb{E}[\mathbf{y}_s\mathbf{y}_s^T]^T+\boldsymbol{\Lambda}\mathbf{U}\mathbb{E}[\mathbf{y}_s\mathbf{y}_s^T]\ ,$$

and that $\mathbf{\Lambda} = \mathbf{\Lambda}^T$, the gradient of $\mathbf{Q}_2$ with respect to $\mathbf{U}$ is:

$$\nabla_{\mathbf{U}} Q_2(\boldsymbol{\theta}, \widetilde{\boldsymbol{\theta}}) = \sum_s \mathbf{\Lambda} \left[\left(\sum_{i=1}^{k(s)} f(\boldsymbol{\phi}_{i,s}, \boldsymbol{\vartheta}) \right) \mathbb{E}\left[\mathbf{y}_s^T\right] - k(s)\mathbf{U}\mathbb{E}\left[\mathbf{y}_s \mathbf{y}_s^T\right] \right] . \tag{35}$$

The specific derivatives and gradients for the affine transformation (3), and for the SAS function (4), can be effectively obtained as detailed in Section IV-A of [25]. These gradients are passed as arguments, together with the objective function, to a Limited memory Broyden–Fletcher–Goldfarb–Shanno (L–BFGS) optimizer [31]–[34] for obtaining the parameters that maximize the log–likelihood of the development set. The L–BFGS optimization is iterated until the log–likelihood of the development data stops improving.
The steps required for estimating the objective function and the gradients with respect to the non-linear function parameters are summarized in Algorithm 1 of [25].

It is interesting noting that, since the gradients with respect to the transformation parameters $\boldsymbol{\vartheta}$ are similar to the ones derived for the speaker–independent non–linear i–vector transformation, NL–PLDA model training can be performed by means of the same forward–backward steps illustrated in Algorithm 1 of [25].
The unique difference is that in [25] we targeted a speaker–independent standard normal distribution for the latent variable $\mathbf{z}$, i.e., $P_{\mathbf{Z}}(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, whereas from (30) we see that in the NL–PLDA model we have to target a speaker–dependent distribution $P_{\mathbf{Z}_s}(\mathbf{z}) = \mathcal{N}\left(\mathbf{z}; \mathbf{U}\mathbb{E}[\mathbf{y}_s], \mathbf{\Lambda}^{-1}\right)$. The only difference, thus, consists in replacing equation (16) of [25] by:

$$P_{\mathbf{Z}_s}(\mathbf{z}) = \mathcal{N}\left(\mathbf{z}; \mathbf{U}\mathbb{E}[\mathbf{y}_s], \mathbf{\Lambda}^{-1}\right) . \tag{36}$$

The corresponding gradient in equation (17) of [25] becomes:

$$\begin{aligned} D_{\mathbf{x}_n}\left[\log P_{\mathbf{Z}}(\mathbf{x}_n)\right] &= \nabla_{\mathbf{x}_n} \log \mathcal{N}\left(\mathbf{x}_n; \mathbf{U}\mathbb{E}[\mathbf{y}_s], \mathbf{\Lambda}^{-1}\right) \\ &= \mathbf{\Lambda}(\mathbf{U}\mathbb{E}\left[\mathbf{y}_s\right] - \mathbf{x}_n) , \end{aligned} \tag{37}$$

where, in accordance with the notation in [25], $\mathbf{x}_n$ is the i–vector transformed applying the non–linear function $f(\mathbf{x}, \boldsymbol{\vartheta})$, i.e., $\mathbf{x}_n = f(\boldsymbol{\phi}, \boldsymbol{\vartheta})$. Thus, the forward–backward procedure illustrated in Algorithm 1 of [25] can be applied as it is for computing the gradients $\nabla_{\boldsymbol{\vartheta}} Q_2(\boldsymbol{\theta}, \widetilde{\boldsymbol{\theta}})$, just replacing the initialization $\mathbf{b}_n = -\mathbf{x}_n$ in the backward recursion of equation (20) of [25] by:

$$\mathbf{b}_n = \mathbf{\Lambda}(\mathbf{U}\mathbb{E}\left[\mathbf{y}_s\right] - \mathbf{x}_n) . \tag{38}$$

V. NL–PLDA SCORING

The likelihood ratio of a pair of i–vectors in a trial can be computed for the NL–PLDA model, according to (12), as shown in (39). Please notice that this equation corresponds to the computation of the likelihood ratio of the standard PLDA model, but using transformed i–vectors $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$, and the matrices $\mathbf{U}$ and $\mathbf{\Lambda}$ estimated as illustrated in the previous section.

VI. I–VECTOR SCALING

We have shown in Section IV of [25] that length–normalization is crucial for PLDA mainly because it reduces the mismatch between the development and evaluation sets, rather than better meeting the PLDA Gaussian assumption. We also proved that length–normalization is optimal for a linear transformation, but a different utterance–dependent scaling factor is necessary for a non–linear transformation.
I–vector normalization is also crucial for NL–PLDA, but as it will be shown in the section devoted to the experiments, blind length–normalization is not effective.

Assuming that i–vectors are sampled from different distributions, characterized by independent scaling parameters, we define the α–scaled non–linear transformation:

$$g(\boldsymbol{\phi}_i, \alpha_i) = \alpha_i \; \boldsymbol{\phi}_i \tag{40}$$

$$f(\boldsymbol{\phi}_i, \alpha_i, \boldsymbol{\vartheta}) = h(g(\boldsymbol{\phi}_i, \alpha_i), \boldsymbol{\vartheta}) , \tag{41}$$

where h is a non–linear transformation, and α_i is an i–vector dependent positive scaling parameter. It is worth noting that α_i can be considered as the single parameter of a simplified affine transformation located at the beginning of the i–vector transformation chain, as shown in Fig. 3. Seen from the other side of the chain, we consider that an i–vector $\boldsymbol{\phi}_i$ is generated by a random variable whose pdf is described by the transformation:

$$\begin{aligned} \overline{\boldsymbol{\Phi}}_i &= \alpha_i^{-1} h^{-1}(\mathbf{z}, \boldsymbol{\vartheta}) \\ &= g^{-1}\left(h^{-1}(\mathbf{z}, \boldsymbol{\vartheta}), \alpha_i\right) , \end{aligned} \tag{42}$$

where h^{-1} and g^{-1} denote the inverse of f and g with respect to $\boldsymbol{\phi}$, given a value for $\boldsymbol{\vartheta}$ and α_i.

The composition of function g with the non–linear function h acts as an i–vector dependent length normalization, tuned for the i–vector distribution described by function h.

A. Estimation of scaling factors for model training

The scaling factor α_i can be obtained by ML estimation, similarly to the other parameters $\boldsymbol{\vartheta}$ of the non–linear transformation. The training algorithm does not change because the α–scaled model can be interpreted as a NL–PLDA model with an i–vector dependent transform, where all parameters of the transformation, excluding each α_i, are tied across all i–vectors.

The terms required for the forward and backward steps of Algorithm 1 in [25] are:

$$\begin{aligned} g(\mathbf{x}, \alpha_i) &= \alpha_i \mathbf{x} \\ D_{\alpha_i} g(\mathbf{x}, \alpha_i) &= \mathbf{x} \\ D_{\mathbf{x}} g(\mathbf{x}, \alpha_i) &= \alpha_i \mathbf{I} \\ \mathcal{G}(\mathbf{x}, \alpha_i) &= N \log \alpha_i \\ D_{\mathbf{x}} \mathcal{G}(\mathbf{x}, \alpha_i) &= \mathbf{0}^T \\ D_{\alpha_i} \mathcal{G}(\mathbf{x}, \alpha_i) &= \frac{N}{\alpha_i} \end{aligned} \tag{43}$$

where $D_{\mathbf{a}} f(\mathbf{x}, \mathbf{a})$ denotes the matrix of partial derivatives of the components of f with respect to the components of $\mathbf{a}$, $\mathcal{G}(\mathbf{x}, \alpha_i) = \log |D_{\mathbf{x}}(\mathbf{x}, \alpha_i)|$, and $| \cdot |$ denotes the absolute

$$
\begin{aligned}
lr &= \frac{P(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2 | \mathcal{H}_S)}{P(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2 | \mathcal{H}_D)} \\
&= \frac{\int \mathcal{N}(f(\boldsymbol{\phi}_1); \mathbf{U}\mathbf{y}, \boldsymbol{\Lambda}^{-1}) \left|\mathbf{D}_{\boldsymbol{\phi}_1} f(\boldsymbol{\phi}_1)\right| \; \mathcal{N}(f(\boldsymbol{\phi}_2); \mathbf{U}\mathbf{y}, \boldsymbol{\Lambda}^{-1}) \left|\mathbf{D}_{\boldsymbol{\phi}_2} f(\boldsymbol{\phi}_2)\right| \; \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I}) \; d\mathbf{y}}{\int \mathcal{N}(f(\boldsymbol{\phi}_1); \mathbf{U}\mathbf{y}, \boldsymbol{\Lambda}^{-1}) \left|\mathbf{D}_{\boldsymbol{\phi}_1} f(\boldsymbol{\phi}_1)\right| \; \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I}) \; d\mathbf{y} \int \mathcal{N}(f(\boldsymbol{\phi}_2); \mathbf{U}\mathbf{y}, \boldsymbol{\Lambda}^{-1}) \left|\mathbf{D}_{\boldsymbol{\phi}_2} f(\boldsymbol{\phi}_2)\right| \; \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I}) \; d\mathbf{y}} \\
&= \frac{\int \mathcal{N}(f(\boldsymbol{\phi}_1); \mathbf{U}\mathbf{y}, \boldsymbol{\Lambda}^{-1}) \; \mathcal{N}(f(\boldsymbol{\phi}_2); \mathbf{U}\mathbf{y}, \boldsymbol{\Lambda}^{-1}) \; \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I}) \; d\mathbf{y}}{\int \mathcal{N}(f(\boldsymbol{\phi}_1); \mathbf{U}\mathbf{y}, \boldsymbol{\Lambda}^{-1}) \; \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I}) \; d\mathbf{y} \;\; \int \mathcal{N}(f(\boldsymbol{\phi}_2); \mathbf{U}\mathbf{y}, \boldsymbol{\Lambda}^{-1}) \; \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I}) \; d\mathbf{y}}
\end{aligned} \tag{39}
$$

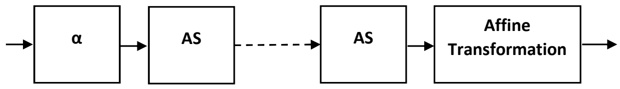

Fig. 3: Chain of transformation functions including the scaling factor module.

value of the determinant of its argument. A matrix of partial derivatives becomes a vector or even a scalar value, if the variable $\mathbf{a}$ or the function value is a scalar.

B. Scoring with NL–PLDA using scaling factors

Differently from PLDA using length normalization, which does not need any additional processing at evaluation time, NL–PLDA with α–scaling must estimate the scaling factors of the trial i-vectors at testing time. In our previous i–vector "gaussianization" approach, α–scaling was applied to each i–vector independently with respect to the others, both in training and in evaluation, because the non–linear transformation was performed before classification.

Due to the introduction of the scaling factors in the NL–PLDA model, the parameters of the non–linear transformation f become i–vector dependent. This makes much more complex the likelihood ratio computation for two main reasons:

1) The parameters α_1 and α_2 of the i–vectors of a trial should be jointly estimated.
2) The terms $\left|\mathbf{D}_{\boldsymbol{\phi}_i} f(\boldsymbol{\phi}_i, \alpha_i)\right|$ cannot be simplified as it was done in (39).

Proper scoring would involve defining an a priori distribution over the scaling factors, and integrating the likelihood of the i–vectors over this prior. Since this computation would be very expensive, we approximate the likelihood ratio computation by optimizing the values of α_1 and α_2 rather than integrating over their prior distribution.

The optimization of α_1 and α_2, however, would require estimating these two parameters under the hypotheses "same–speaker" or "different–speaker", respectively. This would require performing a step of numerical optimization for each trial. Since such approach would be unfeasible for a large set of trials, we further assume that the optimal α_1 and α_2 are the same both for the "same–speaker" and for the "different–speaker" hypotheses. This allows us to independently estimate their values.

Using these approximations, the terms $\left|\mathbf{D}_{\boldsymbol{\phi}_i} f(\boldsymbol{\phi}_i, \alpha_i)\right|$ in the likelihood ratio computation of (39) can again be simplified.

It is worth noting that the estimation of α_1 and α_2 can be performed without resorting to the EM procedure used for training. In particular, the integration over the speaker variable $\mathbf{y}$ can be performed in closed form, before estimating each α_i parameter, thus the computation of the parameter α_i is performed by maximizing:

$$
\begin{aligned}
\alpha_i^* &= \max_{\alpha_i} P(\boldsymbol{\phi}_i | \boldsymbol{\alpha}_i) \\
&= \max_{\alpha_i} \int P_{\boldsymbol{\Phi}|\mathbf{Y}}(\boldsymbol{\phi}_i | \mathbf{y}) P(\mathbf{y}) d\mathbf{y} \\
&= \max_{\alpha_i} \left|\mathbf{D}_{\boldsymbol{\phi}_i} f(\boldsymbol{\phi}_i, \alpha_i)\right| \cdot \\
&\qquad \int \mathcal{N}(f(\boldsymbol{\phi}_i, \alpha_i); \mathbf{U}\mathbf{y}, \boldsymbol{\Lambda}^{-1}) \; \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I}) d\mathbf{y} \\
&= \max_{\alpha_i} \left|\mathbf{D}_{\boldsymbol{\phi}_i} f(\boldsymbol{\phi}_i, \alpha_i)\right| \; \mathcal{N}(f(\boldsymbol{\phi}_i, \alpha_i); \mathbf{0}, \mathbf{U}\mathbf{U}^T + \boldsymbol{\Lambda}^{-1}) \; .
\end{aligned} \tag{44}
$$

The proof that

$$
\begin{aligned}
\int \mathcal{N}(f(\boldsymbol{\phi}_i,\alpha_i); \mathbf{U}\mathbf{y}, \boldsymbol{\Lambda}^{-1}) \; \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I}) \; d\mathbf{y} = \\
\mathcal{N}(f(\boldsymbol{\phi}_i, \alpha_i); \mathbf{0}, \boldsymbol{\Lambda}^{-1} + \mathbf{U}\mathbf{U}^T) \; .
\end{aligned}
$$

is given in Appendix.

Thus, α_i^* can be obtained exploiting again Algorithm 1 of [25], by initializing its backward step by:

$$
\mathbf{b}_n = -\left(\boldsymbol{\Lambda}^{-1} + \mathbf{U}\mathbf{U}^T\right)^{-1} f(\boldsymbol{\phi}_i, \alpha_i, \boldsymbol{\vartheta}) \; . \tag{45}
$$

In summary, at testing time it is only necessary to independently estimate the parameters α_1^* and α_2^*, and to apply the transformation $h(g(\boldsymbol{\phi}_i, \alpha_i^*), \boldsymbol{\vartheta})$ to each evaluation i–vector. The likelihood ratio can then be computed as in standard PLDA, just using α–scaled and transformed i–vectors.

VII. E–VECTORS

Most of the experiments presented in the next section have been performed using a recently introduced representation of a speech segment, similar to the speaker factors of Joint Factor Analysis (JFA) and to i–vectors, referred to as "e–vector" [26]. This representation is based on the observation that JFA estimates a more informative speaker subspace than the "total variability" i–vector subspace, because the latter is obtained by considering each training segment as belonging to a different speaker. The novelty of our proposal consists in estimating a linear transformation that allows keeping the span of the speaker–specific eigenvoice subspace, but at the same time provides a better prior for i–vector extraction. We have shown in [26] that this subspace can be obtained by adapting

TABLE I: Equal Error Rate and $C_{primary}$ for the core extended NIST SRE 2012 evaluations using different, 1024 Gaussian gender–independent models. Label D refers to a system based on a hybrid DNN/GMM architecture. The average % improvement of $C_{primary}$ with respect to the standard PLDA and to the AS–PLDA systems is shown in the last two columns, respectively.

System name	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		Average $C_{primary}$ % improvement with respect to	
	% EER	$C_{primary}$	% EER	$C_{primary}$	% EER	$C_{primary}$	% EER	$C_{primary}$	% EER	$C_{primary}$	PLDA	AS–PLDA
(1) PLDA	3.22	0.292	2.24	0.286	2.66	0.248	4.42	0.401	2.76	0.324	0.0	-
(2) AS–PLDA	2.88	0.238	2.37	0.266	2.81	0.204	4.64	0.408	2.91	0.307	8.3	0.0
(3) NL–PLDA	2.99	0.242	2.33	0.256	2.63	0.193	4.08	0.371	2.81	0.293	12.6	4.8
(4) PSVM	2.91	0.233	2.24	0.264	2.34	0.190	3.89	0.375	2.78	0.302	12.1	4.1
(3) + (4)	2.73	0.215	1.96	0.235	2.18	0.175	3.58	0.341	2.45	0.271	20.2	13.1
(D1) PLDA	3.43	0.258	1.68	0.233	2.77	0.215	4.14	0.346	1.93	0.268	0.0	-
(D2) AS–PLDA	3.09	0.212	1.69	0.208	3.13	0.188	4.15	0.349	2.01	0.241	9.2	0.0
(D3) NL–PLDA	3.40	0.222	1.65	0.203	3.69	0.185	3.55	0.308	1.96	0.229	13.1	4.3
(D4) PSVM	2.86	0.218	1.48	0.208	2.59	0.185	3.39	0.319	1.78	0.236	11.7	2.7
(D3) + (D4)	3.01	0.200	1.28	0.181	2.98	0.168	3.15	0.281	1.54	0.206	21.5	13.5

TABLE II: Comparison of PLDA and NL–PLDA system performance of a hybrid DNN/GMM architecture, in terms of average % EER, DCF08 and $C_{primary}$, for the core extended NIST SRE 2012 evaluations used without and with length normalization, or with e–vector dependent scaling factors.

System	e–vector normalization	% EER	DCF08	$C_{primary}$	Improvement % EER	Improvement DCF08	Improvement $C_{primary}$
PLDA	No	3,73	0,164	0,347	0.0	0.0	0.0
NL–PLDA	No	3,34	0,152	0,326	10,5%	7,3%	6,1%
PLDA	LN	2,79	0,118	0,264	0.0	0.0	0.0
NL–PLDA	LN	3,38	0,138	0,279	-21,1%	-16,9%	-5,7%
NL–PLDA	alpha–scaling	2,85	0,108	0,229	-2,2%	8,5%	13,3%

TABLE III: Same as Table II, but for the female core extended NIST SRE 2010 evaluation. For these experiments the systems are based on 2048 full–covariance GMMs i–vectors.

System	i–vector normalization	% EER	DCF08	DCF10	Improvement % EER	Improvement DCF08	Improvement DCF10
PLDA	No	2.92	0.146	0.438	0.0	0.0	0.0
NL–PLDA	No	2.41	0.115	0.374	17.5%	21.2%	14.4%
PLDA	LN	2.09	0.111	0.405	0.0	0.0	0.0
NL–PLDA	LN	2.04	0.104	0.362	2.4%	6.3%	10.6%
NL–PLDA	alpha–scaling	1.66	0.087	0.349	20.6%	21.6%	13.8%

the eigenvoice subspace by means of a simple procedure that produces a new variability matrix, $\mathbf{E}$. The procedure considers each training segment as belonging to a different speaker, as it is done in standard i–vector training, but it applies solely Minimum Divergence Estimation (MDE) [7], [35] during the training iterations.

In particular, an eigenvoice matrix $\mathbf{V}$ is trained, in a first step, exactly as matrix $\mathbf{T}$ is, but assuming that the segments of a given speaker belong to a single class, i.e., accumulating the sufficient statistics per speaker, rather than per segment.

In the second step, matrix $\mathbf{E}$ is initialized by $\mathbf{V}$. Then, it is re-estimated considering each training segment as belonging to a different speaker, as it is done for the estimation of matrix $\mathbf{T}$, but applying only MDE iterations.

VIII. Experiments

The performance of the proposed approach has been assessed by means of a set of experiments on the SRE 2012 evaluation dataset [36] using e–vectors. The e–vectors were preferred to i–vectors as features because the former have shown in [26] to produce better results than i–vectors both for PLDA and AS–PLDA, using different systems and classifiers.

The e–vectors for the SRE 2012 evaluation were extracted using a relatively small model and feature set. The feature set consists of 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. The e–vector extractor was based on a gender–independent 1024–component diagonal covariance UBM, 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. We implemented gender–independent PLDA classifiers according to the framework illustrated in [9], and also gender–independent PSVMs [11]. All the experiments were performed using e–vectors with dimension $N = 400$. The e–vector extraction post–processing does not include any dimensionality reduction. We also performed a set of experiments using the hybrid DNN/GMM approach of [3]. In particular, we used the approach and the DNN described in [6], associating 8 Gaussians to each of 128 output units of the DNN.

The aim of all these experiments was to compare the effectiveness of the potentially more accurate NL–PLDA model with respect to the AS–PLDA model, and also with respect to the Pairwise Support Vector Machine (PSVM) model [10], [11]. The results are summarized in Table I, where the recognition accuracy is given in terms of percent Equal Error Rate (EER), and of C_{primary}, the cost function defined in the SRE 2012 evaluation plan [36]. The scores were not normalized. The last two columns of the table show the percentage of the average C_{primary} improvement with respect to the reference PLDA (with length–normalized i–vectors) and AS–PLDA systems. The reference systems are easily identified by the 0.0 value in the corresponding rows, both for the GMM, and for the DNN/GMM approach.

The performance of the baseline G–PLDA system using e–vectors is shown in the first row of Table I. The second row shows the results of our best AS–PLDA approach using a cascade of two AS modules, trained with additional constraints on the transformation parameters (α–AS$_2$ bounded) [25]. Using more than two AS modules decreases system performance. These results are consistent with the ones obtained using i–vectors, reported in [25].

The effect of the e–vector "gaussianization" provided by AS–PLDA is to reduce the slope of the DET curve, with a loss of the EER, but a 8.2% improvement of the average C_{primary}. The joint estimation of the PLDA and of the non–linear transformations parameters is more robust to overfitting when more than one AS module is combined in cascade. This probably happens because the NL–PLDA approach takes into account the speaker information, which is ignored by AS–PLDA. The latter tries to gaussianize the development e–vectors assuming that they are independent. Since this is not the case, better fitting the e–vector distribution does not necessarily correspond to better matching the PLDA assumptions. Thus, the new approach allows combining more than a single AS module in cascade to provide a more accurate model. The results of the NL–PLDA system refer to a model including four AS modules.
Excluding Condition 1, NL–PLDA improves both the EER and C_{primary} in all conditions with respect to the AS–PLDA, achieving an average C_{primary} improvement of 4.8%, and outperforming standard PLDA by 12.6%.
It is worth noting that the PSVM classifier gives better % ERR results than standard and non–linear PLDA models. Since the discriminative and the new generative models give similar C_{primary}, but they are trained with different objective functions, we expected a good degree of complementarity between them, even if they exploit the same set of features. Thus, we combined the scores of the two systems with equal weights. The results, given in the row labeled "(3) + (4)", show that the system combination is able to sensibly improve both the EER and the average C_{primary}. The latter achieves an improvement of 9.3%, 8.7%, 13.1%, and 20.2% with respect to PSVM, NL–PLDA, AS–PLDA and PLDA, respectively.

The proposed transformation is also effective for our hybrid DNN/GMM approach [6], which has much better C_{primary} than the GMM reference system. The DNN/GMM based systems are labeled as "D" in Table I.
Again, our best AS–PLDA approach, which uses two bounded AS modules, gets an average C_{primary} improvement of 9.2% with respect to PLDA.
NL–PLDA provides an additional gain of 4.3% , reaching 13.1% improvement with respect to DNN/GMM PLDA. It is interesting noting that the new generative NL–PLDA approach is able to slightly improve the average C_{primary} of discriminative PSVM classifier as well.
Finally, the performance obtained by combining the scores of the PSVM and NL–PLDA systems with equal weights is shown in the last row of Table I. The combination of these two systems produces a significant gain of 11.1%, 9.7%, 13.5%, and 21.5% with respect to the corresponding PSVM, NL-PLDA, AS–PLDA and PLDA, respectively.

A second set of experiments was performed to assess the benefits of the e–vector dependent scaling factors (α–scaling) with respect to length normalization (LN). The results are summarized in Table II. Without any dataset mismatch compensation (i.e. without LN or α–scaling), NL–PLDA results are better than standard PLDA results, but the improvement is less significant compared with the one provided by PLDA length normalized models. This can be explained considering that a better model of the training data is not necessarily more effective in case of significant mismatch between training and evaluation data.

Adding LN to both models improves the results, however, feeding NL-PLDA with length normalized e-vector is detrimental with respect to the standard PLDA with length-normalization. Since we initialize the AS modules to provide an identity transformation, and NL-PLDA with the parameters of the standard PLDA, respectively, and since the likelihood of the system increases during the NL-PLDA training iterations, we argue that an overfitting effect occurs, which increases the mismatch between development and evaluation distributions. This is not surprising: although LN is optimal under Gaussian assumptions (as it has been proved in [25]), it is not optimal for the non–linear PLDA model. Indeed, jointly estimating the scaling factors and the non–linear transformation parameters we obtain optimal factors that allow NL–PLDA to outperform standard PLDA with LN.

Finally, we report the results of the same comparison performed on the female core extended NIST SRE 2010 evaluations. In these experiments we used the same feature set, but i–vectors rather than e–vectors. We trained a gender–independent 2048–component diagonal full–covariance UBM, estimated with data from NIST SRE 2004–2006.
The i–vector extractor was based on a gender–dependent $\mathbf{T}$ matrix, of rank 400, estimated with the female speakers of NIST SRE 2004–2006, and additionally with the Switchboard II, Phases 2 and 3, and Switchboard Cellular, Parts 1 and 2 datasets. I–vectors were used in these experiments because e–vectors require a large enough number of different speaker segments, which are not readily available on the previously mentioned datasets, mostly for the phone/interview conditions. PLDA and NL–PLDA were trained with the dimension of the speaker subspace matrix $\mathbf{U}$ reduced to 150, and also exploiting the NIST SRE 2008 female data.

Table III summarizes the results, which confirm the ones obtained on the NIST SRE 2012, with the exception that NL–PLDA using LN is also better than the corresponding PLDA model.

IX. CONCLUSIONS

We have presented a generative model that jointly estimates the distribution of the development i–vectors and the PLDA parameters, so that the i–vectors are non–linearly transformed to a new compact representation of a speech segment that makes PLDA classification more effective. The i–vector transformation is modeled by means of a sequence of affine and non–linear functions, the parameters of which are obtained by Maximum Likelihood estimation on the development set. The transformation parameters can be estimated using the same algorithm proposed in [25], just properly changing the transformation target distribution. This approach improves both in theory and in practice our previous proposed model, which aimed at minimizing the deviation of the i–vectors from the normal distribution. It also incorporates the benefits of length normalization by estimating speaker–dependent scaling factors, which have been shown to be essential for reducing the mismatch between the development and evaluation i–vector length distributions. Using this new model we were able not only to improve the performance of the PLDA and AS–PLDA generative models, but also to reach the performance of the PSVM discriminative model. The score fusion of these two models provides an additional 9.7% average C_{primary} improvement with respect to the NL–PLDA model.

APPENDIX

Proposition 1:

$$\int \mathcal{N}(f(\boldsymbol{\phi}_i,\alpha_i);\mathbf{U}\mathbf{y},\boldsymbol{\Lambda}^{-1})\ \mathcal{N}(\mathbf{y};\mathbf{0},\mathbf{I})\ d\mathbf{y} = \mathcal{N}(f(\boldsymbol{\phi}_i,\alpha_i);\mathbf{0},\boldsymbol{\Lambda}^{-1}+\mathbf{U}\mathbf{U}^T)\ .$$

Proof:

Let $k(\mathbf{M}) = \frac{|\mathbf{M}|^{\frac{1}{2}}}{(2\pi)^{\frac{D}{2}}}$ where D is the dimension of $\boldsymbol{\phi}_i$, and let us define, for the sake of notation simplicity, $\mathbf{x} = f(\boldsymbol{\phi}_i,\alpha_i)$. Defining also $\boldsymbol{\Lambda}_y = \mathbf{I}+\mathbf{U}^T\boldsymbol{\Lambda}\mathbf{U}$ and $\boldsymbol{\mu}_y = \boldsymbol{\Lambda}_y^{-1}\mathbf{U}^T\boldsymbol{\Lambda}\mathbf{x}$, in analogy with (15) with $k=1$, and noting that both do not depend on $\mathbf{y}$, we get with a bit of algebraic manipulation:

$$\begin{aligned}
&\int \mathcal{N}(\mathbf{x};\mathbf{U}\mathbf{y},\boldsymbol{\Lambda}^{-1})\ \mathcal{N}(\mathbf{y};\mathbf{0},\mathbf{I})\ d\mathbf{y} = \\
&\frac{k(\boldsymbol{\Lambda})k(\mathbf{I})}{k(\boldsymbol{\Lambda}_y)} e^{-\frac{1}{2}\mathbf{x}^T\boldsymbol{\Lambda}\mathbf{x}+\frac{1}{2}\boldsymbol{\mu}_y^T\boldsymbol{\Lambda}_y\boldsymbol{\mu}_y}\ . \\
&\qquad \int e^{\mathbf{y}^T\boldsymbol{\Lambda}_y\boldsymbol{\mu}_y-\frac{1}{2}\mathbf{y}^T\boldsymbol{\Lambda}_y\mathbf{y}-\frac{1}{2}\boldsymbol{\mu}_y^T\boldsymbol{\Lambda}_y\boldsymbol{\mu}_y}\ k(\boldsymbol{\Lambda}_y)\ d\mathbf{y} = \qquad (46)\\
&\frac{k(\boldsymbol{\Lambda})k(\mathbf{I})}{k(\boldsymbol{\Lambda}_y)} e^{-\frac{1}{2}\mathbf{x}^T\boldsymbol{\Lambda}\mathbf{x}+\frac{1}{2}\mathbf{x}^T\boldsymbol{\Lambda}\mathbf{U}\boldsymbol{\Lambda}_y^{-1}\mathbf{U}^T\boldsymbol{\Lambda}\mathbf{x}}\ . \\
&\qquad \int \mathcal{N}(\mathbf{y};\boldsymbol{\mu}_y,\boldsymbol{\Lambda}_y^{-1})\ d\mathbf{y} = \qquad (47)\\
&k(\boldsymbol{\Lambda}\boldsymbol{\Lambda}_y^{-1}) e^{-\frac{1}{2}\mathbf{x}^T(\boldsymbol{\Lambda}-\boldsymbol{\Lambda}\mathbf{U}\boldsymbol{\Lambda}_y^{-1}\mathbf{U}^T\boldsymbol{\Lambda})\mathbf{x}}\ . \qquad (48)
\end{aligned}$$

Using Woodbury identity [37]:

$$(\mathbf{A}+\mathbf{B}\mathbf{C}\mathbf{D})^{-1} = \mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{B}\big(\mathbf{C}^{-1}+\mathbf{D}\mathbf{A}^{-1}\mathbf{B}\big)^{-1}\mathbf{D}\mathbf{A}^{-1}\ , \qquad (49)$$

it is easy verifying that:

$$\begin{aligned}
\Big(\boldsymbol{\Lambda}^{-1}+\mathbf{U}\mathbf{U}^T\Big)^{-1} &= \boldsymbol{\Lambda}-\boldsymbol{\Lambda}\mathbf{U}\Big(\mathbf{I}+\mathbf{U}^T\boldsymbol{\Lambda}\mathbf{U}\Big)^{-1}\mathbf{U}^T\boldsymbol{\Lambda} \\
&= \boldsymbol{\Lambda}-\boldsymbol{\Lambda}\mathbf{U}\boldsymbol{\Lambda}_y^{-1}\mathbf{U}^T\boldsymbol{\Lambda}\ , \qquad (50)
\end{aligned}$$

which is the precision matrix appearing in (48).

Finally, we have to show that:

$$k(\boldsymbol{\Lambda}\boldsymbol{\Lambda}_y^{-1}) = k\left(\Big(\boldsymbol{\Lambda}^{-1}+\mathbf{U}\mathbf{U}^T\Big)^{-1}\right)\ ,$$

that is:

$$\left|\Big(\boldsymbol{\Lambda}^{-1}+\mathbf{U}\mathbf{U}^T\Big)\right|^{-1} = |\boldsymbol{\Lambda}\boldsymbol{\Lambda}_y^{-1}\ .|$$

Recalling Sylvester determinant lemma [38]:

$$|\mathbf{I}+\mathbf{A}\mathbf{B}| = |\mathbf{I}+\mathbf{B}\mathbf{A}|\ , \qquad (51)$$

and its generalization:

$$|\mathbf{A}+\mathbf{B}\mathbf{C}\mathbf{D}| = |\mathbf{A}|\left|\mathbf{C}^{-1}+\mathbf{D}\mathbf{A}^{-1}\mathbf{B}\right||\mathbf{C}|\ , \qquad (52)$$

we have:

$$\begin{aligned}
\left|\boldsymbol{\Lambda}^{-1}+\mathbf{U}\mathbf{U}^T\right|^{-1} &= \Big(\left|\boldsymbol{\Lambda}^{-1}\right|\left|\mathbf{I}+\mathbf{U}^T\boldsymbol{\Lambda}\mathbf{U}\right||\mathbf{I}|\Big)^{-1} \\
&= |\boldsymbol{\Lambda}|\left|\boldsymbol{\Lambda}_y^{-1}\right| = \left|\boldsymbol{\Lambda}\boldsymbol{\Lambda}_y^{-1}\right|\ . \qquad (53)
\end{aligned}$$

Thus, (48) is equal to $\mathcal{N}(\mathbf{x};\mathbf{0},\boldsymbol{\Lambda}^{-1}+\mathbf{U}\mathbf{U}^T)$.

■

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] D. Garcia-Romero, X. Zhang, A. McCree, and D. Povey, “Improving speaker recognition performance in the domain adaptation challenge using deep neural networks,” in *Proceedings of SLT 2014*, pp. 378–383, 2014.

[5] D. Garcia-Romero and A. McCree, “Insights into Deep Neural Networks for speaker recognition,” in *Proceedings of Interspeech 2015*, pp. 1141–1145, 2015.

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

[7] 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.

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

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

[10] S. Cumani, N. Brümmer, L. Burget, P. Laface, O. Plchot, and V. Vasilakakis, “Pairwise discriminative speaker verification in the i-vector space,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 6, pp. 1217–1227, 2013.

[11] S. Cumani and P. Laface, “Large scale training of Pairwise Support Vector Machines for speaker recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 11, pp. 1590–1600, 2014.

[12] J. Rohdin, S. Biswas, and K. Shinoda, “Robust discriminative training against data insufficiency in PLDA-based speaker verification,” *Computer Speech & Language*, vol. 35, pp. 32–7, 2016.

[13] A. Sholokhov, T. Kinnunen, and S. Cumani, "Discriminative multi-domain PLDA for speaker verification," in *Proceedings of ICASSP 2016*, pp. 5030–5034, 2016.
[14] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.
[15] T. Stafylakis, P. Kenny, M. Senoussaoui, and P. Dumouchel, "Preliminary investigation of Boltzmann Machine classifiers for speaker recognition," in *Odyssey 2012*, pp. 109–116, 2012.
[16] M. Senoussaoui, N. Dehak, P. Kenny, R. Dehak, and P. Dumouchel, "First attempt at Boltzmann Machine for speaker recognition," in *Odyssey 2012*, pp. 117–121, 2012.
[17] V. Vasilakakis, S.Cumani, and P. Laface, "Speaker recognition by means of Deep Belief Networks," in *Proceedings Biometric Technologies in Forensic Science (BTFS) 2013*, 2013.
[18] 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.
[19] O. Ghahabi and J. Hernando, "Restricted boltzmann machine supervectors for speaker recognition," in *Proceedings of ICASSP 2015*, pp. 4804–4808, 2015.
[20] D. Snyder, D. Garcia-Romero, and D. Povey, "Time delay Deep Neural Network-based Universal Background Models for speaker recognition," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 92–97, 2015.
[21] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of CVPR 2014*, pp. 1701–1708, 2014.
[22] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *Proceedings of ICASSP 2016*, pp. 5115–5119, 2016.
[23] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *Proceedings of SLT 2016*, 2016.
[24] S. Cumani and P. Laface, "I–vector transformation and scaling for PLDA based speaker recognition," in *Proceedings of Odyssey 2016*, pp. 39–46, 2016.
[25] S. Cumani and P. Laface, "Non–linear i–vector transformations for PLDA based speaker recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 4, pp. 908–919, 2017.
[26] S. Cumani and P. Laface, "E–vectors: JFA and i–vectors revisited," in *Proceedings of ICASSP 2017*, pp. 5435–5439, 2017.
[27] M. C. Jones and A. Pewsey, "Sinh–arcsinh distributions," *Biometrika*, vol. 96, no. 4, pp. 761–780, 2009.
[28] J. F. Rosco, M. C. Jones, and A. Pewsey, "Skew t distributions via the sinh-arcsinh transformation," *TEST*, vol. 20, no. 3, pp. 630–652, 2011.
[29] D. J. Poirier, *Intermediate statistics and econometrics: a comparative approach*. MIT Press, 1995.
[30] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," 2012. Available at http://www2.imm.dtu.dk/pubdb/p.php?3274.
[31] C. G. Broyden, "The convergence of a class of double rank minimization algorithms: 2. the new algorithm," *IMA Journal of Applied Mathematics*, vol. 6, no. 3, pp. 222–231, 1970.
[32] R. Fletcher, "A new approach to variable metric algorithms," *Computer Journal*, vol. 13, no. 3, pp. 317–322, 1970.
[33] D. Goldfarb, "A family of variable metric methods derived by variational means," *Mathematics of Computation*, vol. 24, no. 109, pp. 23–26, 1970.
[34] D. F. Shanno, "Conditioning of quasi-newton methods for function minimization," *Mathematics of Computation*, vol. 24, no. 11, pp. 647–650, 1970.
[35] N. Brummer, "EM for probabilistic LDA," 2010. Technical report, Agnitio Research, Cape Town, Available at sites.google.com/site/nikobrummer/.
[36] "The NIST year 2012 speaker recognition evaluation plan." Available at "http://www.nist.gov/itl/iad/mig/upload/NIST_SRE12_evalplan-v17-r1.pdf.
[37] G. H. Golub and C. V. Loan, *Matrix Computations*. Johns Hopkins, 1996.
[38] D. Harville, *Matrix algebra from a statistician's perspective*. Springer, 2008.

Sandro Cumani received the M.S. degree in Computer Engineering from the Politecnico di Torino, Torino, Italy, in 2008, and the Ph.D. degree in Computer and System Engineering of Politecnico di Torino in 2011. He worked at the Brno University of Technology, Czech Republic, and is a Research Fellow within the Department of Control and Computer Engineering of Politecnico di Torino. His current research interests include machine learning, speech processing and biometrics, in particular speaker and language recognition.

Pietro Laface received the M.S. degree in Electronic Engineering from the Politecnico di Torino, Torino, Italy, in 1973.

Since 1988 it has been full Professor of Computer Science at the Dipartimento di Automatica e Informatica of Politecnico di Torino, where he leads the speech technology research group. He has published over 150 papers in the area of pattern recognition, artificial intelligence, and spoken language processing. His current research interests include all aspects of automatic speech recognition and its applications, in particular speaker and spoken language recognition.