Memory and computation trade-offs for efficient i-vector extraction

Sandro Cumani and Pietro Laface

***Abstract*—This work aims at reducing the memory demand of the data structures that are usually pre–computed and stored for fast computation of the i-vectors, a compact representation of spoken utterances that is used by most state–of–the–art speaker recognition systems. We propose two new approaches allowing accurate i-vector extraction but requiring less memory, showing their relations with the standard computation method introduced for eigenvoices, and with the recently proposed fast eigen–decomposition technique. The first approach computes an i–vector in a Variational Bayes (VB) framework by iterating the estimation of one sub–block of i–vector elements at a time, keeping fixed all the others, and can obtain i–vectors as accurate as the ones obtained by the standard technique but requiring only 25% of its memory. The second technique is based on the Conjugate Gradient solution of a linear system, which is accurate and uses even less memory, but is slower than the VB approach. We analyze and compare the time and memory resources required by all these solutions, which are suited to different applications, and we show that it is possible to get accurate results greatly reducing memory demand compared with the standard solution at almost the same speed.**

***Index Terms*—Speaker Recognition, Eigenvoices, Joint Factor Analysis, i-vectors, Variational Bayes, Conjugate Gradient.**

I. Introduction

Speaker recognition technology has shown continuous improvement in the last decade as confirmed by the results of a series of progressively challenging NIST evaluations [1], [2], and is rapidly moving from research laboratory evaluations to real applications. The scale of these applications ranges from large speaker identification systems, requiring clusters of servers, to simple and fast speaker verification systems to be hosted in mobile devices, where memory and computation resources are limited.

State–of–the–art systems are still based on Gaussian Mixture Models (GMMs), first proposed in [3], [4]. In this approach, a speaker model is represented by a supervector stacking the GMM means, which is adapted from a Universal Background Model (UBM) using Maximum a Posteriori Adaptation. This basic framework, however, has evolved and many efforts have been devoted to robust speaker model adaptation techniques with a limited amount of data, and to devising different solutions to the problem of intersession compensation. In particular, eigenvoice modeling, a technique that constrains the variability of speaker utterances to a low dimensional space, introduced in [5] for speech recognition, has been the inspiration for modern speaker recognition systems. It has proven to be effective for speaker adaptation not only in speech [5], [6] and speaker recognition [7], [8], but also for intersession compensation through eigenchannel modeling [9], [10]. All these approaches rely on Factor Analysis (FA), which allows a compact representation of a speaker or channel model to be obtained as a point in a low–dimensional subspace. A more effective technique that faces intra-speaker variability is Joint Factor Analysis (JFA) [11], [12]. JFA uses the same FA algorithms but jointly estimates speaker and channel variability.

A simpler model for speaker recognition has been introduced in [13], [14], which gets rid of the distinction between speaker and channel variability subspaces, and models both in a common low dimensional space, referred to as the "total variability space". In this approach, a speech segment is represented by a low-dimensional "identity vector" (i-vector for short) extracted by 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 supervector space of the GMM means. Good performance has been obtained using i-vectors and simple LDA and cosine distance scoring [13], but better performance has been achieved by using generative models based on Probabilistic Linear Discriminant Analysis (PLDA) [15], [16]. The goal of such systems is to model the underlying distribution of the speaker and channel components of the i–vectors in a Bayesian framework. From these distributions it is possible to evaluate the likelihood ratio between the "same speaker" hypothesis and "different speakers" hypothesis for a pair of i–vectors. The same paradigm can be used to train discriminative systems where the observation patterns are pairs of i–vectors [17], [18].

In this paper we propose two new approaches allowing accurate i-vector estimation but requiring less memory, showing their relations with the standard method introduced for eigenvoices [6] and with the eigen–decomposition technique [19], which is fast, but inaccurate. The first approach computes an i–vector in a Variational Bayes (VB) framework by iterating the estimation of one sub–block of i–vector elements at a time, keeping fixed all the others. It is able to obtain i–vectors as accurate as the ones obtained by the standard technique but requires only 25% of its memory, running at almost the same speed. The second technique, instead, solves the same linear system by means of the Conjugate Gradient (CG) algorithm, which iteratively refines a complete i–vector. This solution does not require computing and inverting the posterior distribution precision matrix $\mathbf{L}_{\mathcal{X}}$, thus reducing the high storage demands of the standard solution and the costs for the computation and inversion of $\mathbf{L}_{\mathcal{X}}$. The CG technique is accurate and uses even less memory, but is slower than the

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).

VB approach.

We analyze the time and memory costs of all these solutions, which are suited to different application fields, depending on their memory constraints, and we show that it is possible to get accurate results greatly reducing memory demand compared with the standard solution at almost the same speed. It is worth noting that we are mainly concerned with memory costs because the incidence of the time spent for i–vector computation is negligible compared to the importance of keeping the original accuracy and saving memory. While this is true for systems using large models and scoring long speaker segments, real applications often deal with short segments, and constrain the dimensions of the features, of the subspace and the number of Gaussian components that can be used. In these conditions the effectiveness of the i–vector extractor may become more relevant.

This work revises and extends the experiments of [20], reporting preliminary results that have been presented during the Odyssey 2012 workshop. In the same workshop the work [21] has been presented, which uses the VB approach for i–vector extraction. This work introduces two interesting topics with the aim of improving system performance: the extraction of high dimensional i–vectors, made possible by the VB methods, and their usage during training. Our work, instead, was mainly devoted to solve memory and computational issues in recognition, using the popular 400 dimension i–vectors and "standard" or even smaller sized models. Being focused on recognition, we devised and evaluated a VB solution computing one sub–block of i–vector elements at a time, rather than a single element, in order to find a reasonable time/memory trade-off. Moreover, we explore in Section VI-A an issue that is interesting from an application perspective: using the "standard" i–vectors for training the "a–priori knowledge" (the LDA–WCCN or PLDA parameters) and the i–vectors extracted by the VB approach in enrollment and test.

The paper is organized as follows: Section II summarizes the i–vector model for speaker recognition, setting the background for i–vector computation. Section III recalls a recently proposed approximate i–vector estimator approach, which significantly reduces memory demand and processing time. Section IV shows that a Variational Bayes approach, which computes the i–vector components iteratively, converges to the standard solution, thus producing the same i–vectors if enough iterations are performed. The same results can be obtained by another method illustrated in Section V, which is slower but uses less memory. The experimental results are presented and discussed in Section VI, and conclusions are drawn in Section VII.

II. I–VECTOR MODEL

The i–vector model [13], [14] constrains the GMM supervector $\mathbf{s}$, representing both the speaker and channel characteristics of a given speech segment, to live in a single subspace according to:

$$\mathbf{s} = \mathbf{m} + \mathbf{T}\mathbf{w} \ , \tag{1}$$

where $\mathbf{m}$ is the UBM supervector, $\mathbf{T}$ is a low-rank rectangular matrix with $C \times F$ rows and M columns, and C and F are the number of GMM components and feature dimensions, respectively. The M columns of $\mathbf{T}$ are vectors spanning the "total variability" space, and $\mathbf{w}$ is a random vector of size M having a standard normal prior distribution.

Following [11] and the notation in [19], given a sequence of feature vectors $\mathcal{X} = \mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_\tau$ extracted for a speech segment, the corresponding i–vector $\mathbf{w}_\mathcal{X}$ is computed as the mean of the normal posterior distribution $\mathrm{p}(\mathbf{w}|\mathcal{X})$:

$$\mathbf{w}_\mathcal{X} = \mathbf{L}_\mathcal{X}^{-1}\mathbf{T}^*\mathbf{\Sigma}^{-1}\mathbf{f}_\mathcal{X} \ , \tag{2}$$

where $\mathbf{L}_\mathcal{X}$ is the precision matrix of the posterior distribution:

$$\mathbf{L}_\mathcal{X} = \mathbf{I} + \sum_{c=1}^{C} N_\mathcal{X}^{(c)}\mathbf{T}^{(c)^*}\mathbf{\Sigma}^{(c)^{-1}}\mathbf{T}^{(c)} \ . \tag{3}$$

In these equations, $N_\mathcal{X}^{(c)}$ are the zero–order statistics estimated on the c-th Gaussian component of the UBM for the set of feature vectors $\mathcal{X}$, $\mathbf{\Sigma}^{(c)^{-1}}$ is the UBM c–th precision matrix, $\mathbf{\Sigma}$ is the block diagonal matrix with $\mathbf{\Sigma}^{(c)}$ entries, $\mathbf{T}^{(c)}$ is the $F \times M$ sub-matrix of $\mathbf{T}$ corresponding to the c–th mixture component such that $\mathbf{T} = \left(\mathbf{T}^{(1)*}, \ldots, \mathbf{T}^{(C)*}\right)^*$, and $\mathbf{f}_\mathcal{X}$ is the supervector stacking the first–order statistics $\mathbf{f}_\mathcal{X}^{(c)}$, centered around the corresponding UBM means:

$$N_\mathcal{X}^{(c)} = \sum_{t=1}^{\tau} \gamma_t^{(c)} \tag{4}$$

$$\mathbf{f}_\mathcal{X}^{(c)} = \sum_{t=1}^{\tau} \left(\gamma_t^{(c)}\mathbf{x}_t\right) - N_\mathcal{X}^{(c)}\mathbf{m}^{(c)} \ , \tag{5}$$

where $\mathbf{x}_t$ is the t–th feature vector in $\mathcal{X}$, and $\gamma_t^{(c)}$ is its occupation probability.

Since Cholesky decomposition can be applied to each UBM precision matrix $\mathbf{\Sigma}^{(c)^{-1}}$, its contribution can be distributed on its adjacent factors in (2) by setting:

$$\begin{aligned} \mathbf{f}_\mathcal{X}^{(c)} &\leftarrow \mathbf{\Sigma}^{(c)-\frac{1}{2}}\mathbf{f}_\mathcal{X}^{(c)} \\ \mathbf{T}^{(c)} &\leftarrow \mathbf{\Sigma}^{(c)-\frac{1}{2}}\mathbf{T}^{(c)}. \end{aligned} \tag{6}$$

Using these "normalized" statistics and sub–matrices, the i–vector espression in (2) can be written as:

$$\mathbf{w}_\mathcal{X} = \mathbf{L}_\mathcal{X}^{-1}\mathbf{T}^*\mathbf{f}_\mathcal{X} \ , \tag{7}$$

with

$$\mathbf{L}_\mathcal{X} = \mathbf{I} + \sum_{c=1}^{C} N_\mathcal{X}^{(c)}\mathbf{T}^{(c)^*}\mathbf{T}^{(c)} \ . \tag{8}$$

A. Complexity analysis

The complexity of a single i–vector computation mainly depends on the computation of $\mathbf{L}_\mathcal{X}$ and on its inversion. In particular, the computation complexity is $O(M^3 + CFM)$ for (7) plus $O(CFM^2)$ for (8). Usually the number of Gaussian components C is greater than the subspace dimension M, and the latter is greater that the feature dimension F. Popular settings for state–of–the-art systems are: $F = 60$, $C = 2048$, and $M = 400$.

The term $O(CFM^2)$ (quadratic in M) accounts for most of the computation complexity, whereas the memory demand for storing matrix $\mathbf{T}$ is $O(CFM)$.

A faster solution for (8) can be obtained if every term $\mathbf{T}^{(c)^*}\mathbf{T}^{(c)}$ of each mixture component is pre–computed and stored. The computational cost becomes $O(CM^2)$, but the speed–up comes at the expense of an additional (large) memory demand for computing (8), which being $O(CM^2)$ dominates the other memory costs.

In the following we will refer to the latter as the standard method of i–vector extraction, or fast baseline approach, whereas the former will be referred to as the slow baseline approach.

In the next sections we will present a number of approaches aiming at reducing the complexity of the i–vector extraction process either trading accuracy for speed and memory, or even achieving accurate results with less resources. All these approaches focus on the reduction of the storage cost of matrix $\mathbf{L}_\mathcal{X}$, either devising a suitable approximation $\hat{\mathbf{L}}_\mathcal{X}$, as illustrated in Section III and IV, or avoiding altogether its computation as shown in Section V-B, where we rely on the computation of the product $\mathbf{L}_\mathcal{X}\mathbf{w}_k$, which requires much less storage.

III. Approximate i–vector extraction

In this section we assume that the statistics and the $\mathbf{T}^{(c)}$ matrices are normalized according to (6). Since $\mathbf{T}^{(c)^*}\mathbf{T}^{(c)}$ is symmetric and semi–definite positive, it can be eigen–decomposed as:

$$\mathbf{T}^{(c)^*}\mathbf{T}^{(c)} = \mathbf{G}^{(c)}\mathbf{D}^{(c)}\mathbf{G}^{(c)^*} \ , \tag{9}$$

where $\mathbf{G}^{(c)}$ is an orthogonal matrix, and matrix $\mathbf{D}^{(c)}$ is diagonal. $\mathbf{D}^{(c)}$ can be expressed as:

$$\mathbf{D}^{(c)} = \mathbf{G}^{(c)^*}\mathbf{T}^{(c)^*}\mathbf{T}^{(c)}\mathbf{G}^{(c)} \ . \tag{10}$$

A simultaneous orthogonal transformation of the matrices $\mathbf{T}^{(c)}$ has been introduced in [19] for fast computation of the i–vectors with low memory resources, where each $\mathbf{G}^{(c)}$ is replaced, for the sake of efficiency, by a single matrix $\mathbf{Q}$ (see (16) as an example of such a matrix) as:

$$\tilde{\mathbf{D}}^{(c)} = \mathbf{Q}^*\mathbf{T}^{(c)^*}\mathbf{T}^{(c)}\mathbf{Q}. \tag{11}$$

Due to this approximation, every $\tilde{\mathbf{D}}^{(c)}$ will have small non–null off–diagonal elements. Setting to zero these off–diagonal elements diagonalizes $\tilde{\mathbf{D}}^{(c)}$. From (11) we get the approximate matrix:

$$\widetilde{\mathbf{T}^{(c)^*}\mathbf{T}^{(c)}} = \mathbf{Q}\tilde{\mathbf{D}}^{(c)}\mathbf{Q}^* \ , \tag{12}$$

which can be substituted in (8) to obtain the approximated posterior distribution precision matrix:

$$\tilde{\mathbf{L}}_\mathcal{X} = \mathbf{I} + \sum_{c=1}^{C} N_\mathcal{X}^{(c)}\mathbf{Q}\tilde{\mathbf{D}}^{(c)}\mathbf{Q}^* \ . \tag{13}$$

Since $\mathbf{Q}$ is an orthonormal matrix,

$$\tilde{\mathbf{L}}_\mathcal{X} = \mathbf{Q}\hat{\mathbf{L}}_\mathcal{X}\mathbf{Q}^* \tag{14}$$

is a rotated version of the diagonal matrix:

$$\hat{\mathbf{L}}_\mathcal{X} = \mathbf{I} + \sum_{c=1}^{C} N_\mathcal{X}^{(c)}\tilde{\mathbf{D}}^{(c)} \ . \tag{15}$$

Thus, assuming that $\tilde{\mathbf{D}}^{(c)}$ in (11) is diagonal, i.e., that the off-diagonal entries of matrix $\mathbf{D}^{(c)}$ can be ignored, has the remarkable advantage that $\hat{\mathbf{L}}_\mathcal{X}$ is a diagonal matrix that can be computed accumulating C vectors of dimension M and whose inversion cost is negligible.

A suitable common orthogonalizing matrix $\mathbf{Q}$ has been proposed in [19], based on the eigen-decomposition

$$\mathbf{W} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} \tag{16}$$

of the weighted average covariance matrix

$$\mathbf{W} = \sum_{c=1}^{C} \omega^{(c)}\mathbf{T}^{(c)^*}\mathbf{T}^{(c)}, \tag{17}$$

where $\omega^{(c)}$ is the weight of the c–th GMM in the UBM supervector.

By combining (7) and (14), the i–vector $\hat{\mathbf{w}}_\mathcal{X}$ approximating $\mathbf{w}_\mathcal{X}$ is obtained as:

$$\hat{\mathbf{w}}_\mathcal{X} = \mathbf{Q}\hat{\mathbf{L}}_\mathcal{X}^{-1}\mathbf{Q}^*\mathbf{T}^*\mathbf{f}_\mathcal{X} \ . \tag{18}$$

A. Complexity analysis

Using this approach, the computational complexity for the i–vector extraction is reduced to $O(CFM)$, due to $\mathbf{T}^*\mathbf{f}_\mathcal{X}$ in (18). This cost dominates because computing the diagonal matrix $\hat{\mathbf{L}}_\mathcal{X}$ has complexity $O(MC)$, and its inversion is just $O(M)$. The contribution $O(M^2)$ for $\hat{\mathbf{L}}_\mathcal{X}$ back–rotation is also negligible compared to $O(CFM)$. The main contribution to memory costs is $O(CFM)$ for storing matrix $\mathbf{T}$, but additional memory, $O(CM)$ and $O(M^2)$, is needed for storing $\hat{\mathbf{L}}_\mathcal{X}$ and $\mathbf{Q}$, respectively. These additional costs, however, are relatively small because $CF >> M$.

This approach is very fast and memory effective. Its performance is good, as shown in [19], and confirmed in our section devoted to the experiments, but it does not reach the accuracy of the standard approach. Thus we studied alternative memory-aware accurate i–vector extraction methods. In the next section we present the first one: a Variational Bayes approach (VB) that computes i–vectors as accurate as the ones obtained by the standard technique but requires only a fraction of its memory.

IV. Variational Bayes accurate i–vector extraction

Considering the training data $\mathcal{X}$ of a specific speaker, and given the model represented in (1), the joint log-probability of $\mathcal{X}$ and $\mathbf{w}$, according to the notation of Theorem 1 in [11], is given by:

$$\log P_{\mathbf{T},\mathbf{\Sigma}}(\mathcal{X},\mathbf{w}) = \log P_{\mathbf{T},\mathbf{\Sigma}}(\mathcal{X}|\mathbf{w}) + \log P(\mathbf{w}) = \tag{19}$$
$$G_\mathbf{\Sigma} + \mathbf{w}^*\mathbf{T}^*\mathbf{\Sigma}^{-1}\boldsymbol{f}_\mathcal{X} - \frac{1}{2}\mathbf{w}^*\mathbf{T}^*\mathbf{N}_\mathcal{X}\mathbf{\Sigma}^{-1}\mathbf{T}\mathbf{w} - \frac{1}{2}\mathbf{w}^*\mathbf{w} \ ,$$

where $\mathbf{N}_\mathcal{X}$ is a $CF \times CF$ block diagonal matrix with diagonal blocks $N_\mathcal{X}^{(c)}\mathbf{I}$, with $\mathbf{I}$ an $F \times F$ identity matrix, and

$$G_\mathbf{\Sigma} = \sum_{c=1}^{C}\left[N_\mathcal{X}^{(c)}\log\frac{1}{(2\pi)^{F/2}|\mathbf{\Sigma}^{(c)}|^{1/2}} - \frac{1}{2}\mathrm{tr}\left(\mathbf{\Sigma}^{(c)^{-1}}\mathbf{s}_\mathcal{X}^{(c)}\right)\right] \ ,$$

where the $\mathbf{s}_\mathcal{X}^{(c)}$ are the second order centered statistics, and $G_\mathbf{\Sigma}$ collects the terms that do not depend on $\mathbf{w}$.

The posterior distribution of $\mathbf{w}$ can be expressed in closed form [6] as

$$\mathbf{w}|\mathcal{X} \sim \mathcal{N}(\mathbf{w}_{\mathcal{X}}, \mathbf{L}_{\mathcal{X}}^{-1}) \ ,$$

where $\mathbf{w}_{\mathcal{X}}$ and $\mathbf{L}_{\mathcal{X}}$ are given in (2) and (3), respectively.
A diagonal $\mathbf{L}_{\mathcal{X}}$ would imply that the i–vector components w_i are uncorrelated, and thus their posterior distribution would factorize over the components. In the general case, however, $\mathbf{L}_{\mathcal{X}}$ is a full matrix and the i–vector components w_i are correlated in the posterior. For this reason the complexity of the standard approach is much higher than the eigen–decomposition approach. Thus, we look for a variational approximation of the posterior distribution $q(\mathbf{w}) \approx P_{\mathbf{T},\mathbf{\Sigma}}(\mathbf{w}|\mathcal{X})$ having this factorized form:

$$q(\mathbf{w}) = \prod_{i=1}^{B} q(\mathbf{w}_i) \ ,$$

where each $\mathbf{w}_i$ is a set taken from a partition of $\mathbf{w}$ into B disjoint subsets.

Variational Bayes (VB) methods provide a framework to estimate the distributions $q(\mathbf{w}_i)$ that minimize the Kullback-Liebler divergence between the posterior $P_{\mathbf{T},\mathbf{\Sigma}}(\mathbf{w}|\mathcal{X})$ and its approximation $q(\mathbf{w})$ (see Chapter 10.1.1 in [22]). These estimates are given by:

$$\log q(\mathbf{w}_i) = \mathbb{E}_{j\neq i}\left[\log P_{\mathbf{T},\mathbf{\Sigma}}(\mathcal{X}, \mathbf{w})\right] + \text{const} \qquad (20)$$

where, recalling that $\mathbf{w}$ stacks all the elements $\mathbf{w}_i$, the notation $\mathbb{E}_{j\neq i}[\ldots]$ denotes an expectation with respect to the q distributions over all variables $\mathbf{w}_j$ for $j \neq i$.
Let $\overline{\mathbf{T}}_i$ be the matrix *excluding* a block of columns i from $\mathbf{T}$, and let $\overline{\mathbf{w}}_i$ be the vector *excluding* the corresponding elements from vector $\mathbf{w}$

$$\begin{aligned} \overline{\mathbf{w}}_i &= [\mathbf{w}_1, \ldots \mathbf{w}_{i-1}, \mathbf{w}_{i+1}, \ldots \mathbf{w}_B]^* \\ \overline{\mathbf{T}}_i &= [\mathbf{T}_1, \ldots \mathbf{T}_{i-1}, \mathbf{T}_{i+1} \ldots \mathbf{T}_B] \end{aligned}$$

where the sum of the dimensions of the B blocks is equal to the dimension of the subspace M. Using these definitions, the product $\mathbf{Tw}$ can be written as:

$$\mathbf{Tw} = \overline{\mathbf{T}}_i\overline{\mathbf{w}}_i + \mathbf{T}_i\mathbf{w}_i, \qquad (21)$$

which is valid for every $i = 1, \ldots, B$.
Substituting (21) in (19) and the latter in (20), and collecting the terms that do not depend on $\mathbf{w}$ in a constant, we get:

$$\begin{aligned} \log q(\mathbf{w}_i) = \ & \mathbb{E}_{\overline{\mathbf{w}}_i}\Big[\overline{\mathbf{w}}_i^*\overline{\mathbf{T}}_i^*\mathbf{\Sigma}^{-1}\boldsymbol{f}_X + \mathbf{w}_i^*\mathbf{T}_i^*\mathbf{\Sigma}^{-1}\boldsymbol{f}_X \\ & -\frac{1}{2}\overline{\mathbf{w}}_i^*\overline{\mathbf{T}}_i^*\mathbf{N}_{\mathcal{X},}\mathbf{\Sigma}^{-1}\overline{\mathbf{T}}_i\overline{\mathbf{w}}_i - \mathbf{w}_i^*\mathbf{T}_i^*\mathbf{N}_{\mathcal{X}}\mathbf{\Sigma}^{-1}\overline{\mathbf{T}}_i\overline{\mathbf{w}}_i \qquad (22) \\ & -\frac{1}{2}\mathbf{w}_i^*\mathbf{T}_i^*\mathbf{N}_{\mathcal{X}}\mathbf{\Sigma}^{-1}\mathbf{T}_i\mathbf{w}_i - \frac{1}{2}\overline{\mathbf{w}}_i^*\overline{\mathbf{w}}_i - \frac{1}{2}\mathbf{w}_i^*\mathbf{w}_i\Big] + \text{const} \ . \end{aligned}$$

The terms $-\frac{1}{2}\mathbf{w}_i^*\mathbf{T}_i^*\mathbf{N}_{\mathcal{X}}\mathbf{\Sigma}^{-1}\mathbf{T}_i\mathbf{w}_i$ and $-\frac{1}{2}\mathbf{w}_i^*\mathbf{w}_i$, which are quadratic in $\mathbf{w}_i$, can be collected and rewritten as:

$$-\frac{1}{2}\mathbf{w}_i^*\mathbf{\Lambda}_i\mathbf{w}_i \ ,$$

where $\mathbf{\Lambda}_i$ is defined as:

$$\mathbf{\Lambda}_i = \left(\mathbf{T}_i^*\mathbf{N}_{\mathcal{X}}\mathbf{\Sigma}^{-1}\mathbf{T}_i + \mathbf{I}\right) \ . \qquad (23)$$

The terms $\mathbf{w}_i^*\mathbf{T}_i^*\mathbf{\Sigma}^{-1}\boldsymbol{f}_X$ and $-\mathbf{w}_i^*\mathbf{T}_i^*\mathbf{N}_{\mathcal{X}}\mathbf{\Sigma}^{-1}\overline{\mathbf{T}}_i\mathbb{E}_{\overline{\mathbf{w}}_i}[\overline{\mathbf{w}}_i]$, which are linear in $\mathbf{w}_i$, can be collected as:

$$\begin{aligned} &\mathbf{w}_i^*\mathbf{T}_i^*\mathbf{\Sigma}^{-1}\left(\boldsymbol{f}_{\mathcal{X}} - \mathbf{N}_{\mathcal{X}}\overline{\mathbf{T}}_i\mathbb{E}_{\overline{\mathbf{w}}_i}[\overline{\mathbf{w}}_i]\right) = \\ &\mathbf{w}_i^*\mathbf{T}_i^*\mathbf{\Sigma}^{-1}\left(\boldsymbol{f}_{\mathcal{X}} - \mathbf{N}_{\mathcal{X}}\overline{\mathbf{T}}_i\overline{\boldsymbol{\mu}}_i\right), \end{aligned}$$

where $\overline{\boldsymbol{\mu}}_i$ denotes all the current i–vector means of the q distributions excluding the ones in block i.
Since the log–probability of a Gaussian is:

$$\log\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{\Lambda}^{-1}) = -\frac{1}{2}\mathbf{x}^*\mathbf{\Lambda}\mathbf{x} + \mathbf{x}^*\mathbf{\Lambda}\boldsymbol{\mu} + \text{const} \ ,$$

it can be seen that the distribution of $q(\mathbf{w}_i)$ is Gaussian:

$$q(\mathbf{w}_i) \sim \mathcal{N}(\mathbf{w}_i|\boldsymbol{\mu}_i, \mathbf{\Lambda}_i^{-1})$$

with precision matrix $\mathbf{\Lambda}_i$ in (23), and mean:

$$\boldsymbol{\mu}_i = \mathbf{\Lambda}_i^{-1}\mathbf{T}_i^*\mathbf{\Sigma}^{-1}\left(\boldsymbol{f}_{\mathcal{X}} - \mathbf{N}_{\mathcal{X}}\overline{\mathbf{T}}_i\overline{\boldsymbol{\mu}}_i\right). \qquad (24)$$

Thus, the computation of an i–vector can be performed in a Variational Bayes framework by iterating the estimation of one $\boldsymbol{\mu}_i$ at a time, keeping fixed all the others.
Denoting $\overline{\boldsymbol{f}}_{\mathcal{X},i}$ the first–order statistics centered around the new supervector mean $\mathbf{m} + \overline{\mathbf{T}}_i\overline{\boldsymbol{\mu}}_i$, the Gaussian mean $\boldsymbol{\mu}_i$ can be computed as:

$$\boldsymbol{\mu}_i = \mathbf{\Lambda}_i^{-1}\mathbf{T}_i^*\mathbf{\Sigma}^{-1}\overline{\boldsymbol{f}}_{\mathcal{X},i} \ . \qquad (25)$$

A. Fast implementation

It is worth noting that a naive implementation of (25) with a block size $b = 1$ would make the complexity of this approach $O(CFM^2K)$, where K is the number of performed iterations. This is because the computation of $\overline{\boldsymbol{f}}_{\mathcal{X},i}$ would be almost as expensive as computing $\mathbf{T}^*\mathbf{\Sigma}^{-1}\boldsymbol{f}_{\mathcal{X}}$ in (2) and would be repeated for each i–vector dimension at every iteration. However, an efficient implementation is possible by defining and updating a vector $\mathbf{f}_c$ that stores the first order statistics centered around the current supervector mean, defined as:

$$\mathbf{f}_c = \boldsymbol{f}_{\mathcal{X}} - \mathbf{N}_{\mathcal{X}}\sum_{j=1}^{i-1}\mathbf{T}_j\boldsymbol{\mu}_j^{k+1} - \mathbf{N}_{\mathcal{X}}\sum_{j=i}^{B}\mathbf{T}_j\boldsymbol{\mu}_j^{k} \ , \qquad (26)$$

where $\boldsymbol{\mu}_i^k$ refers to the estimate of $\boldsymbol{\mu}_i$ at the k–th iteration. Initially, thus,

$$\mathbf{f}_c = \boldsymbol{f}_{\mathcal{X}} - \mathbf{N}_{\mathcal{X}}\sum_{j=1}^{B}\mathbf{T}_j\boldsymbol{\mu}_j^0 = \boldsymbol{f}_{\mathcal{X}} - \mathbf{N}_{\mathcal{X}}\mathbf{T}\boldsymbol{\mu}^0 \ .$$

Performing the iterations one block at a time from block $i = 1$ to B, defining $\mathbf{K}_i = \mathbf{\Lambda}_i^{-1}\mathbf{T}_i^*\mathbf{\Sigma}^{-1}$, and taking into account (26) the i-th component of the new i–vector is computed at iteration k as:

$$\begin{aligned} \boldsymbol{\mu}_i^{k+1} &= \mathbf{K}_i\left(\boldsymbol{f}_{\mathcal{X}} - \mathbf{N}_{\mathcal{X}}\overline{\mathbf{T}}_i\overline{\boldsymbol{\mu}}_i^k\right) \\ &= \mathbf{K}_i\left(\boldsymbol{f}_{\mathcal{X}} - \mathbf{N}_{\mathcal{X}}\sum_{j=1}^{i-1}\mathbf{T}_j\boldsymbol{\mu}_j^{k+1} - \mathbf{N}_{\mathcal{X}}\sum_{j=i+1}^{B}\mathbf{T}_j\boldsymbol{\mu}_j^{k}\right) \\ &= \mathbf{K}_i\left(\mathbf{f}_c + \mathbf{N}_{\mathcal{X}}\mathbf{T}_i\boldsymbol{\mu}_i^k\right) \ . \qquad (27) \end{aligned}$$

Since $\boldsymbol{\mu}_i$ changes, we must update the vector of the centered first order statistics according to the new $\boldsymbol{\mu}$ simply excluding

the contribution of $\boldsymbol{\mu}_i^k$ and including the one given by the new $\boldsymbol{\mu}_i^{k+1}$, as:

$$\mathbf{f}_c \leftarrow \mathbf{f}_c + \mathbf{N}_{\mathcal{X}}\mathbf{T}_i\boldsymbol{\mu}_i^k - \mathbf{N}_{\mathcal{X}}\mathbf{T}_i\boldsymbol{\mu}_i^{k+1} \ . \tag{28}$$

Notice that $\mathbf{f}_c$ is updated after the computation of each i-th block of the i–vector, not at each iteration k, this is one of the reasons for computing a block rather than a single element of $\mathbf{w}$ at a time.

It is also worth noting that the normalization (6), getting rid of the UBM precision matrices, can be used also in this approach.

B. Complexity analysis

We analyze the computation complexity of the VB approach as a function of the block size b, for the naive and the fast implementation, accounting also the cost of computing $\boldsymbol{\Lambda}_i^{-1}$.

Let's first set aside the computational cost of $\boldsymbol{\Lambda}_i$. The naive VB implementation takes $O(KCF(M-b)M/b)$ for computing the factor $\boldsymbol{f}_{\mathcal{X}} - \mathbf{N}_{\mathcal{X}}\overline{\mathbf{T}}_i\overline{\boldsymbol{\mu}}_i$ in (24), plus $O(KCFM)$ for its product by $\mathbf{T}_i^*\boldsymbol{\Sigma}^{-1}$, and $O(KbM)$ for the product of $\boldsymbol{\Lambda}_i^{-1}$ by the other factors. The fast VB implementation, instead, has complexity $O(KCFM)$ for updating the current first order statistics of (28), plus $O(KCFM)$ for updating (27) and again $O(KbM)$ for performing the final matrix product of $\boldsymbol{\Lambda}_i^{-1}$ with the other factors.

Since (23) has the same form of (8), $\boldsymbol{\Lambda}_i$ can be computed as it has been done for $\mathbf{L}_{\mathcal{X}}$: either accepting the slow solution, which performs the sum of the matrix products to save memory, or by pre–computing and storing the covariance matrices ${\mathbf{T}_i^{(c)}}^*\boldsymbol{\Sigma}^{(c)^{-1}}\mathbf{T}_i^{(c)}$ of each block i for speeding-up the computation. The complexity of the slow solution, which does not require any additional memory, is $O(CFMb)$. It reduces to $O(CMb)$ for the fast solution, but with an additional $O(CMb)$ memory cost. The inversion of all $\boldsymbol{\Lambda}_i$ requires Mb^2 operations.

The minimum memory cost would be obtained by computing a single component at a time, i.e., using a block size $b = 1$. It is necessary, however, to trade the memory occupation and the computational load because the selected block size affects the performance of the matrix multiplication routines, partially the number of iterations necessary for convergence to a preset tolerance value, and also the number of updates of the $\mathbf{f}_c$ centered statistics.

The computation complexity of the VB approach is slightly greater than the standard method, not because it requires a large number of iterations (K), but mostly due to the costs of centering the first order statistics. In Section VI, illustrating the experiments, we will show that very few iterations are required by the VB algorithm to compute suitable i–vectors, obtaining the same performance of the standard approach using about 25% of its memory.

C. Accuracy

In order to show that the expected values $\mathbb{E}[\mathbf{w}_i]$ converge to the standard solution, we rewrite equation (24), using (21) as:

$$\mathbb{E}[\mathbf{w}_i] = \boldsymbol{\Lambda}_i^{-1}\mathbf{T}_i^*\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{f}_{\mathcal{X}} - \boldsymbol{N}_{\mathcal{X}}\mathbf{T}\mathbb{E}[\mathbf{w}] + \boldsymbol{N}_{\mathcal{X}}\mathbf{T}_i\mathbb{E}[\mathbf{w}_i]) \ .$$

Multiplying both sides by $\boldsymbol{\Lambda}_i$ and rearranging the terms we obtain:

$$\mathbf{T}_i^*\boldsymbol{N}_{\mathcal{X}}\boldsymbol{\Sigma}^{-1}\mathbf{T}\mathbb{E}[\mathbf{w}] + \left(\boldsymbol{\Lambda}_i - \mathbf{T}_i^*\boldsymbol{N}_{\mathcal{X}}\boldsymbol{\Sigma}^{-1}\mathbf{T}_i\right)\mathbb{E}[\mathbf{w}_i] = \mathbf{T}_i^*\boldsymbol{\Sigma}^{-1}\boldsymbol{f}_{\mathcal{X}} \ ,$$

and replacing $\boldsymbol{\Lambda}_i$, given by (23), we finally get:

$$\mathbf{T}_i^*\boldsymbol{N}_{\mathcal{X}}\boldsymbol{\Sigma}^{-1}\mathbf{T}\mathbb{E}[\mathbf{w}] + \mathbb{E}[\mathbf{w}_i] = \mathbf{T}_i^*\boldsymbol{\Sigma}^{-1}\boldsymbol{f}_{\mathcal{X}} \ .$$

Thus, the optimal values for the set of $\boldsymbol{\mu}_i = \mathbb{E}[\mathbf{w}_i]$ are given by the solution of the linear system:

$$\left(\mathbf{T}^*\boldsymbol{N}_{\mathcal{X}}\boldsymbol{\Sigma}^{-1}\mathbf{T} + \mathbf{I}\right)\mathbf{w}_{\mathcal{X}} = \mathbf{T}^*\boldsymbol{\Sigma}^{-1}\boldsymbol{f}_{\mathcal{X}} \ , \tag{29}$$

where $\mathbf{T}$ is the matrix that stacks all $\mathbf{T}_i$, and $\mathbf{w}_{\mathcal{X}}$ is the vector that stacks all $\mathbb{E}[\mathbf{w}_i]$.

The i–vector is thus obtained from (29) as:

$$\mathbf{w}_{\mathcal{X}} = \left(\mathbf{T}^*\boldsymbol{N}_{\mathcal{X}}\boldsymbol{\Sigma}^{-1}\mathbf{T} + \mathbf{I}\right)^{-1}\mathbf{T}^*\boldsymbol{\Sigma}^{-1}\boldsymbol{f}_{\mathcal{X}} \ , \tag{30}$$

which corresponds, in matrix form, to the mean of the full posterior $P_{\mathbf{T},\boldsymbol{\Sigma}}(\mathbf{w}|\mathcal{X})$ given in (2) and (3).

V. Solving a linear system

Since in (29) we have a linear system of equations, of form $\mathbf{L}_{\mathcal{X}}\mathbf{w}_{\mathcal{X}} = \mathbf{c}$, it is interesting to analyze its solutions by using standard techniques. There are many iterative algorithms for solving linear systems, but in this work we considered only two of them: the Gauss–Seidel and the Conjugate Gradient (CG). These methods share the property of convergence to the correct solution for positive definite matrices. This condition is satisfied in our system because $\mathbf{L}_{\mathcal{X}}$ is a symmetric and positive definite matrix because it is the sum of $\mathbf{T}^*\boldsymbol{N}_{\mathcal{X}}\boldsymbol{\Sigma}^{-1}\mathbf{T}$, which is a symmetric and positive semidefinite covariance matrix, and of the identity matrix.

The first method has been considered because it shows that the Variational Bayes approach illustrated in the previous section is a generalization of the Gauss–Seidel method, whereas the Conjugate Gradient method has been implemented because it allows obtaining accurate results with the same memory cost of the slow baseline approach.

A. Gauss–Seidel solution

The Gauss-Seidel method can be used to solve iteratively a linear system of equations $\mathbf{L}\mathbf{w} = \mathbf{c}$ element–wise as:

$$w_i^{k+1} = \frac{1}{l_{ii}}\left[c_i - \sum_{j=1}^{i-1} l_{ij}w_j^{k+1} - \sum_{j=i+1}^{B} l_{ij}w_j^k\right], i = 1, \ldots, n \ ,$$

until convergence within a predefined threshold has been achieved. Thus, it would solve iteratively the linear system (29) exactly as our Variational Bayes approach does, just using M blocks of size $b = 1$, but without the speed–up given by the technique illustrated in Section IV-A.

Gauss–Seidel is a special case of the Successive Over Relaxation (SOR) method, where the Gauss-Seidel solution at iteration $k+1$ is linearly combined with the solution of the previous iteration through a factor α as:

$$w^{(k+1)} = \alpha w_{gs}^{(k+1)} + (1-\alpha)w^{(k)} \ .$$

For a symmetric and positive definite matrix the SOR iteration is guaranteed to converge for any value of $0 < \alpha < 2$. The choice of α affects the convergence rate, but finding an optimal value of α is too expensive. Thus, we did not try to heuristically improve the convergence behavior of the VB approach by finding a suitable α value. No preconditioning was necessary because good performance is obtained after just 2 or 3 iterations.

B. Conjugate Gradient solution

Since matrix $\mathbf{L}_{\mathcal{X}}$ is symmetric and positive definite, the linear system of equations (29) can also be solved by the Conjugate Gradient (CG) method, which solves $\mathbf{L}\mathbf{w} = \mathbf{c}$ iterating from an initial guess $\mathbf{w}_0$ and generating successive vectors that are closer to the solution $\mathbf{w}$ that minimizes the quadratic function:

$$f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^*\mathbf{L}\mathbf{w} - \mathbf{w}^*\mathbf{c} \ . \tag{31}$$

Our main interest in this approach comes from the consideration that the iteration updates in this algorithm are based on the residual:

$$\mathbf{r}_k = \mathbf{c} - \mathbf{L}\mathbf{w}_k \ . \tag{32}$$

It is thus possible to reduce the high storage demands of the standard solution and the costs due to the computation and inversion of matrix $\mathbf{L}_{\mathcal{X}}$, because $\mathbf{L}_{\mathcal{X}}$ appears in the residual multiplied by $\mathbf{w}_k$, thus we can avoid the computation of $\mathbf{L}_{\mathcal{X}}$, and rely on the computation of the product $\mathbf{L}_{\mathcal{X}}\mathbf{w}_k$, which requires much less storage. The product

$$\mathbf{L}_{\mathcal{X}}\mathbf{w}_k = (\mathbf{T}^*\boldsymbol{N}_{\mathcal{X}}\boldsymbol{\Sigma}^{-1}\mathbf{T})\mathbf{w}_k + \mathbf{I}\mathbf{w}_k \tag{33}$$

can be computed, right–to–left, by the sequence of operations:

$$\begin{aligned} \mathbf{Z} &= \mathbf{T}\mathbf{w}_k \\ \mathbf{Z} &\leftarrow \boldsymbol{N}_{\mathcal{X}}\boldsymbol{\Sigma}^{-1}\mathbf{Z} \\ \mathbf{Z} &\leftarrow \mathbf{T}^*\mathbf{Z} \\ \mathbf{L}_{\mathcal{X}}\mathbf{w}_k &= \mathbf{Z} + \mathbf{w}_k \ . \end{aligned} \tag{34}$$

The order of the operations is important because the first operation produces a vector, which is then scaled by the values of the diagonal matrix $\boldsymbol{N}_{\mathcal{X}}\boldsymbol{\Sigma}^{-1}$, and finally $\mathbf{L}_{\mathcal{X}}\mathbf{w}_k$ is obtained by the last two operations. It is worth noting that even if $\boldsymbol{\Sigma}^{-1}$ in (33) were a full matrix, it could be distributed to the adjacent factors $\mathbf{T}^*$ and $\mathbf{T}$ as has been done in (6).

C. Complexity analysis

The computation complexity of the CG method, using this approach, is $O(KCFM)$ for the first and third operation, plus $O(KCF)$ for the second and $O(KM)$ for the fourth one in (34). The cost of a single CG iteration is less expensive compared to the standard fast computation, which is $O(CFM^2)$. Although few iterations are usually necessary to reach an acceptable approximation, the Conjugate Gradient approach is not as fast as the standard approach, but it uses far less memory, i.e., the same memory required by the baseline slow approach to keep in memory matrix $\mathbf{T}$ ($O(CFM)$).

VI. Experiments

Since the focus of this work was i–vector extraction, we did not devote particular care to select the best combination of features, techniques, and training data that allow obtaining the best performance. However, we targeted the parameters of state–of–the-art systems, i.e., large feature and model dimensions, and good results, in order to avoid biased conclusions that could not be extended to the best recognition systems. For the same reason, we tested the techniques introduced in the previous sections only on the female part of the tel-tel extended NIST 2010 evaluation trials [23], using two classifiers having the same front–end, based on cepstral features.

In particular, we extracted, every 10 ms, 19 Mel frequency cepstral coefficients and the frame log-energy on a 25 ms sliding Hamming window. This 20–dimensional feature vector was subject to short time mean and variance normalization using a 3 s sliding window, and a 60-dimensional feature vector was obtained by appending the delta and double delta coefficients computed on a 5–frame window. We trained a gender-independent UBM, modeled by a diagonal covariance 2048-component GMM, and also a gender-independent $\mathbf{T}$ matrix using only the NIST SRE04 SRE05 and SRE06 datasets. The i-vector dimension was fixed to 400 for all the experiments.

The first recognition system that has been tested is based on the LDA–WCCN approach [14], which performs intersession compensation by means of Linear Discriminant Analysis (LDA), where all the i-vectors of the same speaker are associated with the same class. LDA removes the nuisance directions from the i-vectors by reducing the feature dimensions (from 400 to 200 in our tests, as in the original proposal [14]). These speaker features are finally normalized by means of Within Class Covariance Normalization (WCCN) [24], and used for cosine distance scoring (CS). The second system is based on Gaussian PLDA, implemented according to the framework illustrated in [15]. We trained models with full–rank channel factors, using 120 dimensions for the speaker factors.

The LDA matrix, the WCCN transformations, and the PLDA models have been trained using the NIST SRE04, SRE05, SRE06 datasets, and additionally the Switchboard II, Phases 2 and 3, and Switchboard Cellular, Parts 1 and 2 datasets. The i–vectors of the PLDA models are L_2 normalized, whereas the scores provided by both systems are not normalized. All the experiments were first performed by using i–vectors extracted by the same technique. Thus, the i–vectors extracted for training the LDA–WCCN or the PLDA parameters and the enrollment and test i–vectors were consistent. We also devised a second set of experiments for evaluating the accuracy of the systems considering the possibility that the i–vectors used for LDA–WCCN or the PLDA training were the "standard" ones, whereas the enrollment and test i–vectors were not consistent. In particular, it is interesting from an application viewpoint to train once the a–priori knowledge (the LDA–WCCN or PLDA parameters) using the standard approach, and to use a different i–vector extractor in enrollment and testing due to memory or computational constraints. We evaluated two possible scenarios keeping fixed the a–priori

knowledge: in the first one, both enrollment and test i–vectors are extracted by one of the non–standard approaches, whereas just the test i–vectors are not consistent in the second scenario.

Table I summarizes the performance of the evaluated approaches on the female part of the extended telephone condition in the NIST 2010 evaluation. The results reported in this table were obtained using the same i–vector extractor for enrollment, test and a–priori knowledge training. The recognition accuracy is given in terms of percent Equal Error Rate (% EER) and Minimum Detection Cost Functions defined by NIST for the 2008 (minDCF08$\times$1000) and 2010 (minDCF10$\times$1000) evaluations [23], respectively.

It is worth noting that preliminary results about memory and computational costs were reported in [20], which cannot be compared with the one given in Table I because in that work we had not considered some straightforward memory and computational optimizations that can be applied both to the "standard" i–vector extractor and to the proposed techniques. In particular, we report here the results obtained using single rather than double precision floating-point storage because we have verified that the performance of the system, in any condition, is not affected by single precision operations. Using single rather than double precision floating-point immediately halves the required storage. The second optimization that allows saving both memory and time consists of storing only the lower triangular elements of the symmetric matrices. These optimizations implied also a revision of our previous findings about the relative computational costs of some of the devised i–vector extraction implementations.

In Table I, label VB–b–θ refers to the Variational Bayes approach with block size b and stopping criterion threshold equal to θ. The stopping criterion is based on the difference between the L_2-norm of the current estimated i–vector and the one computed in the previous iteration. In label CG–θ, instead, θ indicates the residual norm (32) stopping criterion threshold of the Conjugate Gradient approach.

Analyzing the results in Table I it is evident that, no matter the i–vector extraction technique used, the accuracy of the PLDA system is significantly better than the LDA-WCCN cosine distance scoring approach.

The fast baseline results, corresponding to the standard i–vector extraction approach, are obtained 18 times faster than the corresponding slow approach. However, the latter requires only 188 MB for storing matrix $\mathbf{T}$, whereas the former needs 4 times more memory, essentially to store the terms $\mathbf{T}^{(c)^*}\mathbf{T}^{(c)}$ that allow speeding–up the computation of (8).

The approximate i–vector extraction based on eigen-decomposition of Section III is extremely fast and requires almost the same amount of memory as the accurate slow approach. However, it is not able to reach the accuracy of the baseline system. On the contrary, in Section IV-C it has been shown that the VB approach, given enough iterations, converges to the i–vectors of the standard solution, thus it gives the same results of the standard system. This can be appreciated looking at the results reported in the lines labeled VB–b–10, corresponding to a tight threshold and slower extraction times with respect to the faster VB–b–100 systems.

As illustrated in Section IV-B, the block size b in the VB approach affects memory and computation costs. Thus, we performed a set of experiments evaluating the performance of these systems using different values of the block size. System accuracy does not show significant variations for a given stopping threshold. However, looking also at Figure 1, which shows the computational time plotted against the required memory as a function of the block sizes, it can be seen that the best speed and memory trade–off is obtained by selecting a block size $b = 20$. Moreover, for the sake of efficiency, the iterations can be terminated before the convergence to the "standard" i–vector has been reached using a less restrictive stopping criterion. Table I and Figure 1 show that using a larger stopping criterion threshold $\theta = 100$ no appreciable performance degradation is observed with the "approximate" i–vectors obtained in about half processing time. Thus, the accuracy of the system is not particularly affected by VB i–vectors that significantly differ from the "standard" ones.

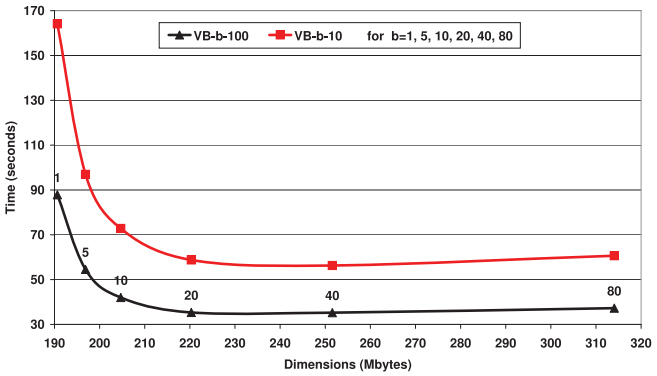


Fig. 1: Computation time plotted against memory dimension, as a function of block sizes: 1, 5, 10, 20, 40, and 80, respectively.

Summarizing the Variational Bayes approach results, we conclude that the VB–20–10 system is able to get the same results of the baseline systems, 1.6 to 2.3 times slower than the standard approach, depending on the available number of concurrent threads, but using slightly more memory than the memory efficient slow baseline system, or the fast, but inaccurate, eigen–decomposition approach. Very good performance is also obtained by the VB–20–100 system, with an earlier stop of the iterations, leading to i–vector extraction, which is at worst 1.4 times slower than the standard method when using a single thread, but requires only 1/4 of its memory.

Two threshold values have also been tested for the Conjugate Gradient approach to evaluate how the threshold value affects the recognition accuracy. The results, given at the bottom of Table I, show that the stopping criterion is not critical. The CG approach is slower than the standard and VB methods. However, it achieves the same accuracy using the same amount of memory of the slow baseline approach, just the one needed for storing matrix $\mathbf{T}$, even slightly less than the Eigen–decomposition approach. Faster convergence is obtained preconditioning the algorithm by initializing the precision matrix $\mathbf{L}_{\mathcal{X}}$ as the diagonal of the weighted average

TABLE I: Results for the extended NIST SRE2010 female tests in terms of % EER, minDCF08×1000 and minDCF10×1000 using different i–vector extraction approaches, and consistent i–vectors

System	Memory	1 core **200** utterances 2.2M frames		12 cores **1000** utterances 12M frames		Cosine Scoring			PLDA		
						(%) EER	min DCF08	min DCF10	(%) EER	min DCF08	min DCF10
	(MB)	time ratio	cpu time	time ratio	cpu time						
Fast baseline	814	1.0	26.09 s	1.0	39.97 s	5.00	229	614	3.59	181	568
Slow baseline	188	18.2	474.69 s	9.2	369.71 s	5.00	229	614	3.59	181	568
Eigen–decompo	191	0.3	6.72 s	0.2	7.5 s	5.70	252	692	4.26	202	690
VB-1-10	191	6.3	164.14 s	4.4	176.02 s	4.96	229	613	3.51	181	566
VB-5-10	197	3.7	96.95 s	2.0	79.46 s	4.96	228	618	3.48	181	565
VB-10-10	205	2.8	72.81 s	1.7	68.05 s	4.94	230	619	3.56	181	567
VB-20-10	220	2.3	58.77 s	1.6	63.48 s	4.94	229	621	3.51	182	570
VB-40-10	252	2.2	56.27 s	1.6	63.39 s	4.97	230	614	3.56	181	567
VB-80-10	314	2.3	60.65 s	1.6	65.35 s	4.95	230	615	3.54	179	564
VB-1-100	191	3.4	87.84 s	2.1	84.12 s	5.15	229	620	3.67	183	593
VB-5-100	197	2.1	54.55 s	1.2	47.83 s	5.24	233	615	3.59	184	584
VB-10-100	205	1.6	42.00 s	1.0	40.07 s	5.16	233	627	3.62	183	590
VB-20-100	220	1.4	35.34 s	1.0	39.62 s	5.16	232	622	3.51	183	573
VB-40-100	252	1.4	35.24 s	1.0	38.01 s	5.10	232	621	3.62	181	573
VB-80-100	314	1.4	37.25 s	1.1	41.97 s	5.13	231	622	3.68	181	575
CG-10	188	6.0	155.94 s	3.1	124.21 s	5.00	229	616	3.61	180	568
CG-100	188	3.7	95.4 s	2.1	83.28 s	5.20	224	620	3.62	182	568
CG-10 Preconditioned	191	4.2	109.71 s	2.4	95.90 s	4.97	229	615	3.59	181	566
CG-100 Preconditioned	191	2.7	70.85 s	1.7	68.79 s	5.13	234	620	3.51	185	579

covariance matrix:

$$\mathbf{L}_{\mathcal{X}} = \text{diag} \sum_{c=1}^{C} (N_{\mathcal{X}}^{(c)} \mathbf{T}^{(c)^*} \mathbf{T}^{(c)}) + \mathbf{I} \ . \tag{35}$$

Preconditioning speeds up the CG approach by 30% approximately.

A. Results with not consistent i–vectors

The results of the second set of experiments, where the LDA–WCCN or the PLDA parameters were trained using the "standard" i–vectors, whereas the enrollment and test i–vectors were not consistent, are given in Table II. The performance of the systems is reported, as in the previous table, in terms of % EER, and minDCF08×1000, and minDCF10×1000. In the first tested system configuration, both enrollment and test i–vectors are extracted by one of the non–standard approaches. In the second configuration only the test i–vectors are not consistent with respect to the standard extraction approach. The fast baseline results are, of course, identical to the ones in Table I.

A comparison among these results and the ones obtained using consistent i–vectors is summarized in the bar graphs of Figure 2. In these graphs, each group of bars refers to an i–vector extraction approach, and each bar corresponds to a classification method (labels CS and PLDA refer to Cosine Scoring and PLDA classification, respectively), and enrollment and test configuration. The first bar in each group is the reference performance of a CS system using consistent i–vectors, whereas CS∼E∼T and CS∼T show the performance of a system using non–standard i-vectors both in enrollment and test, or non–standard i-vectors only in testing, respectively. The same information is given by the other three bars in each group, for PLDA systems.

The results of these tests using not consistent i–vectors confirm the gaps between the PLDA and LDA–WCCN classification, and among the eigen–decomposition and the other i–vector extraction approaches. In particular, a relevant performance decrease is observed for the CS eigen–decomposition approach using both enrollment and test approximate i–vectors, whereas the approximate VB–b–100 and CS–100 i–vectors do not show appreciable degradation in the same conditions. Better results are obtained by the eigen–decomposition i–vectors with a PLDA classifier, but they are not comparable with the very good results, similar to the baseline approach, which are obtained using the VB–b–100 and CS–100 i–vectors.

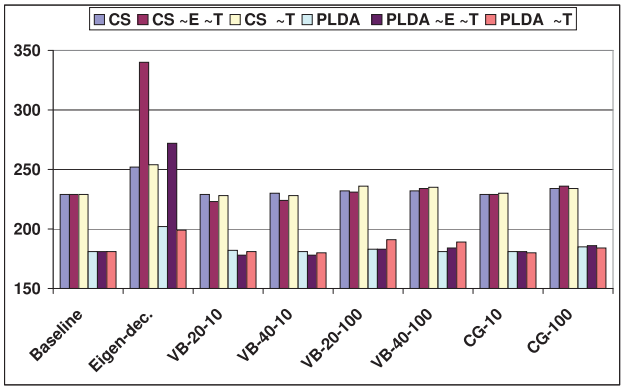
Looking at the performance obtained using non–standard i–vectors only in test, as expected, there is a very small degradation for the VB and CG systems. Surprisingly, looking mostly at the %EER values, it can be seen that the eigen–decomposition i–vectors behave better in this condition than in the previous one. We conclude that the classifiers' performance is significantly affected when both i–vectors are largely approximated, possibly because the two eigen–decomposition i–vectors are approximated in opposite directions with respect to the corresponding standard i–vector, thus the error is reduced if one of the i–vectors is consistent.

B. Relative cost of i–vector extraction

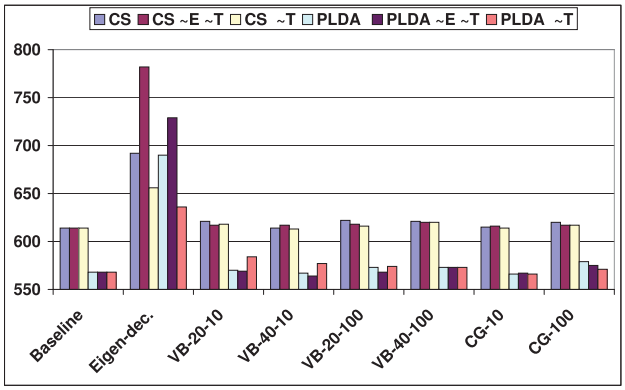
It is worth considering that i–vector extraction is only one of the steps involved in the speaker recognition process. Voice activity detection, feature extraction, Gaussian selection, collection of the zero and first order statistics, i–vector scoring and score normalization are, of course, time consuming modules. Thus, the incidence of the time spent for i–vector computation in a system using large models and scoring long speaker segments is negligible compared to the importance of

TABLE II: Results for the extended NIST SRE2010 female tests in terms of % EER, minDCF08×1000 and minDCF10×1000 using different combinations of i–vector extractors for PLDA training, enrollment and test

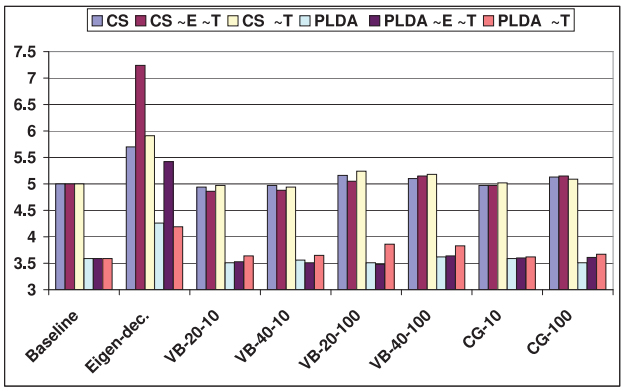
	Non–standard i-vectors in enrollment and test						Non–standard i-vectors only in test					
	Cosine Scoring			PLDA			Cosine Scoring			PLDA		
System	EER	DCF08	DCF10	EER	DCF08	DCF10	EER	DCF08	DCF10	EER	DCF08	DCF10
Fast baseline	5.00	229	614	3.59	181	568	5.00	229	614	3.59	181	568
Eigen–decomposition	7.24	340	782	5.42	272	729	5.91	254	656	4.19	199	636
VB-20-10	4.86	223	617	3.53	178	569	4.97	228	618	3.64	181	584
VB-40-10	4.88	224	617	3.51	178	564	4.94	228	613	3.65	180	577
VB-20-100	5.05	231	618	3.48	183	568	5.24	236	616	3.86	191	574
VB-40-100	5.15	234	620	3.64	184	573	5.18	235	620	3.83	189	573
CG-10 Preconditioned	4.97	229	616	3.60	181	567	5.02	230	614	3.62	180	566
CG-100 Preconditioned	5.15	236	617	3.61	186	575	5.09	234	617	3.67	184	571



(a) minDCF08×1000



(b) minDCF10×1000



(c) % EER

Fig. 2: Comparison of the % EER, minDCF08×1000, and minDCF10×1000 of different system configurations. Labels CS and PLDA refer to Cosine Scoring and PLDA classification, respectively. ∼E∼T and ∼T refer to the extraction of non–standard i-vectors both in enrollment and test, or in test only, respectively.

keeping the original accuracy and saving memory. However, while large models are typically used for NIST evaluations, real applications often constrain the number of Gaussian components, the dimensions of the features, and the size of i–vectors that can be used. Moreover, while the duration of the voice regions in the tel-tel conversations of NIST 2010 is approximately 2 minutes, several applications deal with much shorter segments. Since an i–vector summarizes the speaker information of a speaker segment, the complexity of its extraction does not depend on the length of the segment, thus the effectiveness of the i–vector extractor is more relevant for systems dealing with short utterances such as, for example, the text prompts in speaker verification.

In order to assess the incidence of i–vector extraction time on the overall recognition process time we performed a set of experiments using two i–vector systems with smaller models and shorter segments. In particular, the relative contribution of i–vector extraction to the overall processing time has been measured reducing the feature dimension to 25 MFCC parameters ($c_1 - c_{12}, \Delta c_0 - \Delta c_{12}$), and using 512 or 1024

TABLE III: Percentage of the overall recognition time devoted to i–vector extraction using the standard approach

System	512 G		1024 G	
Segment duration (sec)	30	15	30	15
Relative cost (%)	21	34	34	50

Gaussian components, parameters that are more suited to real applications [25].

For these experiments, we extracted from randomly selected test trials 100 segments of duration 15s and 30s, respectively. The results, reported in Table III, show the percentage of the overall recognition time devoted to i–vector extraction using the standard approach, as a function of the segment duration and of the number of Gaussians. For a given segment duration, the percentage of time spent by i–vector extraction increases, doubling the dimensions of the models because the time spent for Gaussian selection and collection of statistics does not increase as much as the i–vector extraction does. For a given model dimension, the relative cost of i–vector extraction increases for short utterances because the time devoted to i–vector extraction does not depend on the segment duration. However, when both accuracy and memory requirements of the system are a concern, the Variational Bayes approach offers a good solution with a limited increase of the computation times.

VII. Conclusions

The aim of this work was to optimize the memory, and possibly computation time, required for the i-vector extraction module of a speaker recognition system. Although this optimization is particularly useful for small footprint applications, it can be also relevant for speaker identification and verification applications, where the duration of the available speaker segments is short.

We analyzed the time and memory resources required by two new techniques for i-vector extraction. Their implementation has been compared with the standard one, and with the eigen–decomposition approximation. Our approaches, although not as fast as the eigen–decomposition technique, allow obtaining accurate i–vectors and results, and require much less memory than the standard technique.

There are two key ideas in our proposal to cope with memory constraints. The first one, which exploits the Variational Bayes framework, is the iterative optimization of $\boldsymbol{\mu}_i$ in (25) which does not require the full precision matrix $\mathbf{L}_{\mathcal{X}}$, but only its diagonal blocks $\boldsymbol{\Lambda}_i$ in (23), reducing the memory complexity from $O(CM^2)$ to $O(CM(F+b))$, where b is the block size. The second one is the elimination of the computation, and inversion, of the posterior distribution precision matrix $\mathbf{L}_{\mathcal{X}}$, which can can be avoided in the Conjugate Gradient solution because its iterations do not require $\mathbf{L}_{\mathcal{X}}$ but only the product $\mathbf{L}_{\mathcal{X}}\mathbf{w}$, which is far less expensive to compute and store than $\mathbf{L}_{\mathcal{X}}$, as it has been shown in Section V-B.

Using the common settings of most state–of–the–art systems, i.e., 2048 GMMs, 60 dimensional MFCC features, and 400 dimension i–vectors, and the Variational Bayes technique, accurate results were obtained by using a fraction of the memory needed for the standard approach, in almost the same time. Even less memory is required for the Conjugate Gradient approach, which allows obtaining accurate results, but is slower than the standard approach. This drawback is negligible for applications focusing on speaker recognition in conversations.

In summary, the two proposed approaches substantially reduce the memory cost of the i–vector extraction module, which becomes comparable to the eigen–decomposition technique. Both are slower than eigen–decomposition, which is, however, inaccurate. The VB technique is faster that the Conjugate Gradient solution and is the preferred choice, but the latter is preferable in case of strict memory constraints.

VIII. Acknowledgments

We would like to thank Patrick Kenny from CRIM for triggering our interest in the topic of memory efficient i–vector extraction in a private communication, and to Daniele Colibro and Claudio Vair from Loquendo, for useful discussions.

References

[1] M. A. Przybocki, A. F. Martin, and A. N. Le, "NIST Speaker Recognition Evaluations utilizing the Mixer corpora–2004, 2005, 2006," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 1951–1959, 2007.

[2] A. F. Martin and C. S. Greenberg, "The NIST 2010 Speaker Recognition Evaluation," in *Proceedings of INTERSPEECH 2010*, pp. 2726–2729, 2010.

[3] D. A. Reynolds, "Speaker identification and verification using Gaussian Mixture Speaker Models," *Speech Communications*, vol. 17, no. 1-2, pp. 91–108, August 1995.

[4] 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.

[5] R. Kuhn, J. Junqua, P. Nguyen, and N. Niedzielski, "Rapid speaker adaptation in eigenvoice space," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 6, pp. 695–707, 2000.

[6] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE Transactions on Audio Speech and Language Processing*, vol. 15, no. 4, pp. 1435–1447, 2005.

[7] S. Lucey and T. Chen, "Improved speaker verification through probabilistic subspace adaptation," in *Proceedings of EUROSPEECH 2003*, pp. 2021–2024, 2003.

[8] P. Kenny, M. Mihoubi, and P. Dumouchel, "New MAP estimators for speaker recognition," in *Proceedings of EUROSPEECH 2003*, pp. 2964–2967, 2003.

[9] N. Brümmer, "The Spescom Data Voice NIST sre 2004 system," in *NIST SRE 2004 Evaluation Workshop*, 2004. presented at NIST SRE 2004 Evaluation Workshop, Toledo, Spain.

[10] R. Vogt, B. Baker, and S. Sridharan, "Modeling session variability in text-independent speaker verification," in *Proceedings of INTERSPEECH 2005*, pp. 3117–3120, 2005.

[11] P. Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," 2005. Technical report CRIM-06/08-13.

[12] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint Factor Analysis versus eigenchannels in speaker recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 3, pp. 1435–1447, 2005.

[13] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, and P. Ouellet, "Support Vector Machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Proceedings of Interspeech 2009*, pp. 1559–1562, 2009.

[14] 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.

[15] 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.
[16] S. J. D. Prince and J. H. Elder, "Probabilistic Linear Discriminant Analysis for inferences about identity," in *Proceedings of 11th International Conference on Computer Vision*, pp. 1–8, 2007.
[17] L. Burget, O. Plchot, S. Cumani, O. Glembek, P. Matějka, and N. Brümmer, "Discriminatively trained Probabilistic Linear Discriminant Analysis for speaker verification," in *Proceedings of ICASSP 2011*, pp. 4832–4835, 2011.
[18] S. Cumani, N. Brümmer, L. Burget, and P. Laface, "Fast discriminative speaker verification in the i–vector space," in *Proceedings of ICASSP 2011*, pp. 4852–4855, 2011.
[19] O. Glembek, L. Burget, P. Matějka, M. Karafiát, and P. Kenny, "Simplification and optimization of i–vector extraction," in *Proceedings of ICASSP 2011*, pp. 4516–4519, 2011.
[20] S. Cumani, P. Laface, and V. Vasilakakis, "Memory and computation effective approaches for i–vector extraction," in *Proceedings of Odyssey 2012: The Speaker Recognition Workshop*, pp. 9–15, 2012.
[21] P. Kenny, "A small footprint i-vector extractor," in *Proceedings of Odyssey 2012*, pp. 1–7, 2012.
[22] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[23] "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.
[24] A. Hatch, S. Kajarekar, and A. Stolcke, "Within-class covariance normalization for SVM-based speaker recognition," in *Proceedings of ICSLP 2006*, pp. 1471–1474, 2006.
[25] S. Cumani, P. D. Batzu, D. Colibro, C. Vair, P. Laface, and V. Vasilakakis, "Comparison of speaker recognition approaches for real applications," in *Proceedings of Interspeech 2011*, pp. 2365–2368, 2011.

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 2012. He is currently also with Brno University of Technology, Czech Republic. 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 120 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.