Odyssey 2012
The Speaker and Language Recognition Workshop
25-28 June 2012, Singapore

Memory and computation effective approaches for i–vector extraction

Sandro Cumani, Pietro Laface and Vasileios Vasilakakis

Politecnico di Torino, Italy
{Sandro.Cumani, Pietro.Laface, Vasileios.Vasilakakis}@polito.it

Abstract

This paper focuses on the extraction of i-vectors, a compact representation of spoken utterances that is used by most of the state–of–the–art speaker recognition systems.
This work was mainly motivated by the need of reducing the memory demand of the huge data structures that are usually precomputed for fast computation of the i-vectors.
We propose a set of new approaches allowing accurate i-vector extraction but requiring less memory, showing their relations with the standard computation method introduced for eigenvoices. We analyze the time and memory resources required by these solutions, which are suited to different fields of application, and we show that it is possible to get accurate results with solutions that reduce both computation time and memory demand compared with the standard solution.

1. Introduction

Speaker recognition technology has shown relevant improvement in the last decade, 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 [1]. 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 focusing on robust speaker model adaptation techniques with limited amount of data, and proposing 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 [2] for speech recognition, has been the inspiration for modern speaker recognition systems. It has proved to be effective for speaker adaptation not only in speech [2, 3] and speaker recognition [4, 5], but also for intersession compensation through eigenchannel modeling [6, 7]. 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 much more effective technique that faces at the same time intra-speaker variability is Joint Factor Analysis (JFA) [8, 9]. JFA uses the same FA algorithms but jointly estimates speaker and channel variability.

A simpler model for speaker recognition has been introduced in [10, 11], which gets rid of the distinction between speaker and channel variability subspaces, and models both in a common constrained 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 [10], but better performance has been achieved by using generative models based on Probabilistic Linear Discriminant Analysis (PLDA) [12, 13]. 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 [14, 15].

In this paper we propose a set of new approaches allowing accurate i-vector estimation but requiring less memory, showing their relations with the standard method introduced for eigenvoices. We analyze the time and memory costs of these solutions which are suited to different fields of application, and finally we propose a new i-vector estimator technique that is able to obtain accurate results using a fraction of the memory of the standard approach and running three times faster.

The paper is organized as follows: Section 2 summarizes the i–vector model for speaker recognition, setting the background for i–vector computation and model training. Section 3 recalls a recently proposed approximate i–vector estimator approach that significantly reduces memory demand and processing time. Section 4 shows that a Variational Bayes approach, which computes the i–vector components iteratively, ends up to the standard solution in matrix form, thus producing the same i–vectors. The same results can be obtained with the same or less computation time of the standard solution, but with less memory using two methods illustrated in Section 5. The experimental results are presented and commented in Section 6, and conclusions are drawn in Section 7.

2. i–vector model

The i–vector model [10, 11] 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

Vasileios Vasilakakis is supported by the FP7/2007-2013 European Programme under grant agreement n.238803

number of GMM components and feature dimension, 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 [8] and the notation in [16], given a sequence of feature vectors $\mathcal{X} = \mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_t$ extracted for a speech segment, the corresponding i–vector $\mathbf{w}_\mathcal{X}$ is computed as the mean of the 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 $\mathbf{L}$ is the precision matrix of the posterior distribution:

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

In these equations, $N_\mathcal{X}^{(c)}$ are the zero–order statistics estimated on the c-th Gaussian component of the UBM observing the set of feature vectors in $\mathcal{X}$, $\mathbf{\Sigma}^{(c)^{-1}}$ 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:

$$N_\mathcal{X}^{(c)} = \sum_t \gamma_t^{(c)} \tag{4}$$

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

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

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

2.1. 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. The results reported in Section 6.1 have been obtained with settings often used for state–of–the-art systems: $F = 60$, $C = 2048$, 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 $O(CM^2)$ for (8) can be obtained if every covariance matrix $\mathbf{T}^{(c)^*}\mathbf{T}^{(c)}$ is pre–computed and stored for each mixture component. The speed–up comes, however, at the expense of an additional (large) memory demand for computing (8), which dominates the other memory costs because it is $O(CM^2)$.

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 cost of matrix $\mathbf{L}_\mathcal{X}$ computation, either devising a suitable approximation $\hat{\mathbf{L}}_\mathcal{X}$, or avoiding altogether its computation saving memory and time.

3. Approximate i–vector extraction

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 [16] 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 $\mathbf{Q}$.

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

and every $\tilde{\mathbf{D}}^{(c)}$, which is no more diagonal, is forced to be diagonal by setting to zero its off-diagonal elements.
Substituting (11) in (8), we get a diagonalized posterior distribution precision matrix:

$$\hat{\mathbf{L}}_\mathcal{X} = \mathbf{I} + \left(\sum_c N_\mathcal{X}^{(c)}\tilde{\mathbf{D}}^{(c)}\right) \tag{12}$$

and the approximated posterior distribution precision matrix, can be obtained by back–rotation of $\hat{\mathbf{L}}_\mathcal{X}$, as

$$\tilde{\mathbf{L}}_\mathcal{X} = \mathbf{Q}\hat{\mathbf{L}}_\mathcal{X}\mathbf{Q}^* \tag{13}$$

Assuming that $\mathbf{D}^{(c)}$ in (11) is diagonal, i.e. that its off-diagonal entries can be ignored, has the remarkable advantage that $\hat{\mathbf{L}}_\mathcal{X}$ can be computed by C element–wise products of two vectors of dimension M, and its inversion is negligible.

A suitable common orthogonalizing matrix $\mathbf{Q}$ has been proposed in [16], based on the eigen-decomposition

$$\mathbf{W} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} \tag{14}$$

of the weighted average covariance matrix

$$\mathbf{W} = \sum_c \omega^{(c)}\mathbf{T}^{(c)^*}\mathbf{T}^{(c)} \tag{15}$$

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

By combining (7) and (13), the i–vector $\hat{\mathbf{w}}_\mathcal{X}$ approximating $\mathbf{w}_\mathcal{X}$ is obtained as:

$$\hat{\mathbf{w}}_\mathcal{X} = \mathbf{Q}\hat{\mathbf{L}}_\mathcal{X}^{-1}\mathbf{Q}^*\mathbf{T}^*\mathbf{f}_\mathcal{X} \tag{16}$$

3.1. 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 (16). 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 it has been shown in [16], 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 that computes i–vectors as accurate as the ones obtained by the standard technique but requires only a fraction of its memory.

4. 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 $\mathbf{w}$ is given by:

$$\log P_{\mathbf{T},\mathbf{\Sigma}}(\mathcal{X},\mathbf{w}) = \log P_{\mathbf{T},\mathbf{\Sigma}}(\mathcal{X}|\mathbf{w}) + \log P(\mathbf{w}) = \quad (17)$$
$$G_{\mathbf{\Sigma}} + \mathbf{w}^*\mathbf{T}^*\mathbf{\Sigma}^{-1}\boldsymbol{f}_\mathcal{X} - \frac{1}{2}\mathbf{w}^*\mathbf{T}^*\mathbf{N}_\mathcal{X}\mathbf{\Sigma}^{-1}\mathbf{T}\mathbf{w} - \frac{1}{2}\mathbf{w}^*\mathbf{w}$$

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

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

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

The posterior distribution of $\mathbf{w}$ can be expressed in closed form [3] 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 (7) and (8), 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 general case, however, $\mathbf{L}_\mathcal{X}$ is a full rank 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 $\mathbf{w}_i$ is a set taken from a partition of $\mathbf{w}$ into B disjoint subsets.

Variational Bayes (VB) provides 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 $q(\mathbf{w})$ [17]. These estimates are given by

$$\log q(\mathbf{w}_i) = \mathbb{E}_{j\neq i}\left[\log P_{\mathbf{T},\mathbf{\Sigma}}(\mathcal{X},\mathbf{w})\right] + \text{const} \quad (18)$$

Let $\overline{\mathbf{T}}_i$ be the matrix *excluding* a block of adjacent columns i from $\mathbf{T}$, and let $\overline{\mathbf{w}}_i$ be the vector *excluding* the corresponding elements from vector $\mathbf{w}$

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

where the sum of the dimensions of the B blocks is equal to the dimension of the subspace M. The product $\mathbf{T}\mathbf{w}$ can be thus written as:

$$\mathbf{T}\mathbf{w} = \overline{\mathbf{T}}_i\overline{\mathbf{w}}_i + \mathbf{T}_i\mathbf{w}_i \quad (19)$$

that is valid for every $i = 1, \dots, B$.
Substituting (19) in (17) and the latter in (18), and ignoring the terms that do not depend on $\mathbf{w}$, we get:

$$\begin{aligned}\log q(\mathbf{w}_i) = \mathbb{E}_{j\neq i}\Big[&\overline{\mathbf{T}}_i\overline{\mathbf{w}}_i\mathbf{\Sigma}^{-1}\boldsymbol{f}_X + \mathbf{T}_i\mathbf{w}_i\mathbf{\Sigma}^{-1}\boldsymbol{f}_X \\ &-\frac{1}{2}\overline{\mathbf{w}}_i^*\overline{\mathbf{T}}_i^*\mathbf{N}_{\mathcal{X},}\mathbf{\Sigma}^{-1}\overline{\mathbf{T}}_i\overline{\mathbf{w}}_i - \mathbf{w}_i\mathbf{T}_i\mathbf{N}_\mathcal{X}\mathbf{\Sigma}^{-1}\overline{\mathbf{T}}_i\overline{\mathbf{w}}_i \\ &-\frac{1}{2}\mathbf{w}_i^*\mathbf{T}_i^*\mathbf{N}_\mathcal{X}\mathbf{\Sigma}^{-1}\mathbf{T}_i\mathbf{w}_i - \frac{1}{2}\overline{\mathbf{w}}_i^*\overline{\mathbf{w}}_i - \frac{1}{2}\mathbf{w}_i^*\mathbf{w}_i\Big] + \text{const}\end{aligned} \quad (20)$$

which shows that the distribution of $q(\mathbf{w}_i)$ is Gaussian:

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

The terms that are quadratic in $\mathbf{w}_i$ represent its precision matrix:

$$\mathbf{\Lambda}_i = \left(\mathbf{T}_i^*\mathbf{N}_\mathcal{X}\mathbf{\Sigma}^{-1}\mathbf{T}_i + \mathbf{I}\right) \quad (21)$$

and the Gaussian mean $\boldsymbol{\mu}_i$ can be obtained from the terms that are linear in $\mathbf{w}_i$:

$$\begin{aligned}\boldsymbol{\mu}_i &= \mathbf{\Lambda}_i^{-1}\mathbf{T}_i^*\mathbf{\Sigma}^{-1}\left(\boldsymbol{f}_\mathcal{X} - \mathbf{N}_\mathcal{X}\overline{\mathbf{T}}_i\mathbb{E}\left[\overline{\mathbf{w}}_i\right]\right) \quad (22)\\ &= \mathbf{\Lambda}_i^{-1}\mathbf{T}_i^*\mathbf{\Sigma}^{-1}\left(\boldsymbol{f}_\mathcal{X} - \mathbf{N}_\mathcal{X}\overline{\mathbf{T}}_i\overline{\boldsymbol{\mu}}_i\right)\end{aligned}$$

where $\overline{\boldsymbol{\mu}}_i$ denotes all the current i–vector means excluding the ones included in block i.
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:

$$\boldsymbol{\mu}_i = \mathbf{\Lambda}_i^{-1}\mathbf{T}_i^*\mathbf{\Sigma}^{-1}\overline{\boldsymbol{f}}_{\mathcal{X},i} \quad (23)$$

4.1. Fast implementation

It is worth noting that a naive implementation of (23) with a block size equal to 1 would make the complexity of this approach $O(CFM^2K)$, where K is the number of performed iterations. This is because the computation load of $\mathbf{T}_i^*\mathbf{\Sigma}^{-1}\overline{\boldsymbol{f}}_{\mathcal{X},i}$ is $O(CF(M-1))$ that would be almost as expensive as computing $\mathbf{T}^*\boldsymbol{f}_\mathcal{X}$ in (7) and would be repeated for each i–vector dimension at every iteration.

An efficient implementation is, however, possible by defining and updating a vector $\mathbf{f}_c$ that stores the first order statistics centered around the current supervector mean. At the beginning of the iterations the vector is set to:

$$\begin{aligned}\mathbf{f}_c^{old} &= \boldsymbol{f}_\mathcal{X} - \mathbf{N}_\mathcal{X}\mathbf{T}\boldsymbol{\mu}^0 \quad (24)\\ &= \boldsymbol{f}_\mathcal{X} - \sum_{j=1}^{B}\mathbf{T}_j\boldsymbol{\mu}_j^0\end{aligned}$$

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

$$\begin{aligned}\boldsymbol{\mu}_i^{k+1} &= \mathbf{K}_i\left(\boldsymbol{f}_{\mathcal{X}} - \mathbf{N}_{\mathcal{X}}\overline{\mathbf{T}}_i\overline{\boldsymbol{\mu}}_i\right) \\ &= \mathbf{K}_i\left(\boldsymbol{f}_{\mathcal{X}} - \mathbf{N}_{\mathcal{X}}\sum_{j<i}\mathbf{T}_j\boldsymbol{\mu}_j^{k+1} - \mathbf{N}_{\mathcal{X}}\sum_{j>i}\mathbf{T}_j\boldsymbol{\mu}_j^{k}\right) \\ &= \mathbf{K}_i\left(\mathbf{f}_c + \mathbf{N}_{\mathcal{X}}\mathbf{T}_i\boldsymbol{\mu}_i^k\right) \qquad (25)\end{aligned}$$

and the new vector of the centered first order statistics becomes:

$$\begin{aligned}\mathbf{f}_c^{new} &= \boldsymbol{f}_{\mathcal{X}} - \mathbf{N}_{\mathcal{X}}\sum_{j<\leq i}\mathbf{T}_j\boldsymbol{\mu}_j^{k+1} - \mathbf{N}_{\mathcal{X}}\sum_{j>i}\mathbf{T}_j\boldsymbol{\mu}_j^{k} \\ &= \mathbf{f}_c^{old} + \mathbf{N}_{\mathcal{X}}\sum_{j<i}\mathbf{T}_j\boldsymbol{\mu}_j^{k+1} + \mathbf{N}_{\mathcal{X}}\sum_{j\geq i}\mathbf{T}_j\boldsymbol{\mu}_j^{k} - \\ &\quad \mathbf{N}_{\mathcal{X}}\sum_{j\leq i}\mathbf{T}_j\boldsymbol{\mu}_j^{k+1} - \mathbf{N}_{\mathcal{X}}\sum_{j>i}\mathbf{T}_j\boldsymbol{\mu}_j^{k} \\ &= \mathbf{f}_c^{old} + \mathbf{N}_{\mathcal{X}}\mathbf{T}_i\boldsymbol{\mu}_i^k - \mathbf{N}_{\mathcal{X}}\mathbf{T}_i\boldsymbol{\mu}_i^{k+1} \qquad (26)\end{aligned}$$

4.2. Complexity analysis

The computation complexity of the VB approach must be analyzed, as a function of the block size b, for the naive and the fast implementation, and taking also into account the cost of computing $\mathbf{\Lambda}_i^{-1}$.

Let's first set aside the computational cost of $\mathbf{\Lambda}_i$. 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 (22), plus $O(KCFM)$ for its product with $\mathbf{T}_i^*\mathbf{\Sigma}^{-1}$, and $O(KbM)$ for the product of $\mathbf{\Lambda}_i^{-1}$ with the other factors.

The fast VB implementation, instead, has complexity $O(KCFM)$ for updating the current first order statistics of (26), plus $O(KCFM)$ for updating (25) and again $O(KbM)$ for performing the final matrix product of $\mathbf{\Lambda}_i^{-1}$ with the other factors.

Since (21) has the same form of (8), $\mathbf{\Lambda}_i$ 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 $\mathbf{T}_i^{(c)^*}\mathbf{\Sigma}^{(c)^{-1}}\mathbf{T}_i^{(c)}$ of each block i for speeding-up the computation.

The complexity of the slow solution, which does not require any additional memory, is $O(CFMb)$, which reduces to $O(CMb)$ for the fast solution, but with an additional $O(CMb)$ memory cost. The inversion of all $\mathbf{\Lambda}_i$ 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 number of iterations necessary for convergence to a preset tolerance value.

Compared with the standard method, the computation complexity of the fast VB approach is thus reduced from $O(CM^2)$ to $O(KCFM)$ because, as shown in the section devoted to the experiments, the number of iterations required by the VB algorithm to compute suitable i–vectors is small. Overall we will show that the fast VB technique gets the same performance of the standard approach using less than 20% of its memory.

4.3. Accuracy

In order to show that the expected values $\mathbb{E}(\mathbf{w}_i)$ converge to the standard solution, we rewrite equation (22) as:

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

Multiplying both sides by $\mathbf{\Lambda}_i$ and rearranging we obtain:

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

and replacing $\mathbf{\Lambda}_i$ given by (21) we finally get:

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

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

$$\left(\mathbf{T}^*\boldsymbol{N}_{\mathcal{X}}\mathbf{\Sigma}^{-1}\mathbf{T} + \mathbf{I}\right)\mathbf{w}_{\mathcal{X}} = \mathbf{T}^*\mathbf{\Sigma}^{-1}\boldsymbol{f}_{\mathcal{X}} \qquad (27)$$

or

$$\mathbf{w}_{\mathcal{X}} = \left(\mathbf{T}^*\boldsymbol{N}_{\mathcal{X}}\mathbf{\Sigma}^{-1}\mathbf{T} + \mathbf{I}\right)^{-1}\mathbf{T}^*\mathbf{\Sigma}^{-1}\boldsymbol{f}_{\mathcal{X}} \qquad (28)$$

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

5. Solving a linear system

Since (27) is a linear system of equations, of form $\mathbf{L}_{\mathcal{X}}\mathbf{w}_{\mathcal{X}} = \mathbf{c}$, it is interesting to analyze the solutions that can be obtained 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 $\mathbf{L}_{\mathcal{X}} = \mathbf{T}^*\boldsymbol{N}_{\mathcal{X}}\mathbf{\Sigma}^{-1}\mathbf{T} + \mathbf{I}$ is a symmetric and positive definite matrix:it is the sum of $\mathbf{T}^*\boldsymbol{N}_{\mathcal{X}}\mathbf{\Sigma}^{-1}\mathbf{T}$, which is a symmetric and positive semidefinite covariance matrix and of an identity matrix.

The first method has been considered also 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 selected because it is fast and allows obtaining accurate results with less memory.

5.1. Gauss–Seidel solution

The Gauss-Seidel method solves our linear system of equations $\mathbf{L}\mathbf{w} = \mathbf{c}$ element–wise

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

iteratively until convergence within a predefined threshold has been achieved. Thus, it would solve iteratively the linear system (27) 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 4.1.

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.

5.2. Conjugate Gradient solution

Since matrix $\mathbf{L}$ is symmetric and positive definite, the Conjugate Gradient method (CG) method can be used to solve the linear system of equations in (27). It proceeds 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:

$$f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^*\mathbf{L}\mathbf{w} - \mathbf{w}^*\mathbf{c} \tag{29}$$

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

$$\mathbf{r}_k = \mathbf{c} - \mathbf{L}_{\mathcal{X}}\mathbf{w}_k \tag{30}$$

It is thus possible to reduce the i–vector extraction complexity, avoiding the high storage demands of the standard solution and the costs due to the computation and inversion of matrix $\mathbf{L}_{\mathcal{X}}$, because in the residual $\mathbf{L}_{\mathcal{X}}$ appears multiplied by $\mathbf{w}_k$. The product $\mathbf{L}_{\mathcal{X}}\mathbf{w}_k$ can be performed as:

$$\mathbf{L}_{\mathcal{X}}\mathbf{w}_k = (\mathbf{T}^*\boldsymbol{N}_{\mathcal{X}}\boldsymbol{\Sigma}^{-1}\mathbf{T})\mathbf{w}_k + \mathbf{I}\mathbf{w}_k \tag{31}$$

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

$$\begin{aligned} \mathbf{Z} &= \mathbf{T}\mathbf{w}_k \\ \mathbf{Z} &\leftarrow \boldsymbol{N}_{\mathcal{X}}\boldsymbol{\Sigma}^{-1}\mathbf{Z} \\ \mathbf{Z} &\leftarrow \mathbf{T}^*\mathbf{Z} \\ \mathbf{L}_{\mathcal{X}}\mathbf{w}_k &= \mathbf{Z} + \mathbf{w}_k \end{aligned} \tag{32}$$

The order of the operations is important because the first operation gives 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 (31) is a full matrix, it can be distributed to the adjacent factors $\mathbf{T}^*$ and $\mathbf{T}$ as it has been done in (6).

5.3. Complexity analysis

The computation complexity of the GD method thus is $O(KCFM)$ for the first and third operation, plus $O(KCF)$ for the second and $O(M)$ for the fourth one in (32). The cost of a single GC iteration is less expensive compared to the standard fast computation, which is $O(CFM^2)$. Since few iterations are usually necessary to reach an acceptable approximation, the Conjugate Gradient approach is as fast as the standard approach, but it uses far less memory, i.e. the same memory O(CFM) required by the baseline slow approach to keep in memory matrix $\mathbf{T}$.

6. 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 taking biased conclusions that could not be extended to the best recognition systems.

We tested the techniques introduced in the previous sections on the female part of the tel-tel extended NIST 2010 evaluation trials [18], by using two i–vector systems 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 subjected 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 SRE 04/05/06 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 [11], 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 [11]). These speaker features are finally normalized by means of Within Class Covariance Normalization (WCCN) [19], and used for cosine distance scoring.

The second system is based on Gaussian PLDA, implemented according to the framework illustrated in [12]. 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 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.

6.1. Results

Table 1 summarizes the performance of the evaluated approaches on the female part of the extended telephone condition in the NIST 2010 evaluation. The results were obtained using 2048 Gaussian systems.

The recognition accuracy is given in terms of Equal Error Rate (EER) and Minimum Detection Cost Functions defined by NIST for the 2008 (minDCF08) and 2010 (minDCF10) evaluations [18].

The first observation is 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.

Our fast baseline results, corresponding to the standard i–vector extraction approach, are obtained 3–4 time faster than the corresponding slow approach. However, the latter requires only 375 MB for storing matrix $\mathbf{T}$, whereas the former needs 7 times more memory to store the terms $\mathbf{T}^{(c)*}\mathbf{T}^{(c)}$ required to speed–up the computation of (8).

The approximate i–vector extraction based on eigen-decomposition of Section 3 is extremely fast and requires almost the same amount of memory of the accurate slow approach. However, it is not able to reach the accuracy of the baseline system.

In Section 4.3 it has been shown that the Variational Bayes approach, given enough iterations, converges to the i–vectors

Table 1: Results for the extended NIST SRE2010 female tests in terms of % EER, minDCF08 and minDCF10 with different i–vector extractors

System	Memory	1 core **100** utterances 1127224 frames		12 cores **500** utterances 6168082 frames		Cosine Scoring			PLDA		
						(%) EER	min DCF08	min DCF10	(%) EER	min DCF08	min DCF10
	(MB)	rel. speedup	cpu time	rel. speedup	cpu time						
Fast baseline	2875	1	116.8 s	1	109.2 s	4.97	230	612	3.59	180	567
Slow baseline	375	0.27	435.9 s	0.36	306.5 s	4.97	230	612	3.59	180	567
Eigen–decomp.	382	29.30	4.0 s	12.71	8.8 s	5.67	252	697	4.26	202	685
VB-20-100	500	4.23	27.5 s	2.98	37.5 s	5.13	232	622	3.51	183	576
VB-20-10	500	2.63	44.3 s	1.60	69.5 s	4.93	229	621	3.46	182	569
CG-100	375	0.85	136.6 s	1.48	75.4 s	5.16	224	618	3.59	183	567
CG-10	375	0.56	207.4 s	0.99	110.4 s	4.96	230	612	3.59	179	564

of the standard solution, thus it gives the same results of the standard system.

As illustrated in Section 4.2, the block size b affects memory and computation costs. We did experiments with several values of the block size. While the system accuracy does not change, we found that the best speed and memory trade–off was obtained by setting $b = 20$. Moreover, for the sake of efficiency, the iterations can be terminated before the convergence to the "standard" i–vector has been reached, i.e. when the difference between the L_2-norm of the current estimated i–vector and the one computed in the previous iteration is less than a predefined threshold.

The two rows in Table 1 labeled VB-20-100 and VB-20-10 refer to i–vectors extracted according to the Variational Bayes approach with block size $b = 20$, using as stopping criterion a threshold equal to 100 and 10, respectively. The first threshold has been selected to show that the accuracy of the system is not particularly affected by i–vectors that significantly differ from the "'standard' ones. Using 1/3 more memory than the memory efficient slow baseline system, the VB-20-10 system is able to get the same results of the baseline systems 1.6 to 2.6 times faster than the standard approach, depending on the available number of concurrent threads. Very good performance is also obtained by the VB-20-100 system, with an earlier stop of the iterations, leading to a 3 to 4 times faster i–vector extraction.

In the Conjugate Gradient approach the stopping criterion is based on the value of the residual (30). Again two threshold values have been tested to show that the threshold value is not critical for the recognition accuracy. The Conjugate Gradient approach is as fast as the standard method and achieves the same accuracy, but it uses the same amount of memory of the slow baseline approach, just the one required for storing matrix $\mathbf{T}$, even slightly less than the Eigen–decomposition approach.

Table 2: 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

6.2. 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 keeping the original accuracy and saving memory. However, while large models are typically used for NIST evaluations, real applications often constrain the dimensions of the features, of the subspace and the number of Gaussian components 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 depended 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 Gaussian components, parameters that are more suited to real applications [20]. 100 segments of duration 15s and 30s, respectively were extracted from random selected test trials.

The results of these experiments are summarized in Table 2, which shows for the 512 and 1024 Gaussian systems, the percentage of the overall recognition time devoted to i–vector extraction using the standard approach, as a function of the segment duration. 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. In these conditions, a fast technique, such as the VB approach, gives an important contribution to the reduction of the recognition times.

7. 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 are not as fast as the eigen–decomposition technique, but allow obtaining accurate i–vectors and results, and require much less memory than the standard technique.

Two are the key ideas in our proposals. The first one is the fast iterative optimization of $\boldsymbol{\mu}_i$ in (23) and $\boldsymbol{\Lambda}_i$ in (21) in the Variational Bayes technique. The second one is the elimination of the computation, and inversion, of the posterior distribution precision matrix $\mathbf{L}_{\mathcal{X}}$, which is possible in the Conjugate Gradient solution because it does not require $\mathbf{L}_{\mathcal{X}}$ but only $\mathbf{L}_{\mathcal{X}}\mathbf{w}_k$.

Using the settings of most high performance systems, 2048 GMMs, 60 dimensional MFCC features, and 400 dimension i–vectors, and the Variational Bayes technique, the i–vector extractions is three times faster and we obtain accurate results using a fraction of the memory needed for the standard approach. Even less memory is required for the Conjugate Gradient approach, which allows obtaining accurate results, but requires more time than the standard approach. This drawback could be negligible for applications focusing on speaker recognition in conversations.

8. 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.

9. 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] 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.

[3] 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.

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

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

[6] P. Kenny, M. Mihoubi, and P. Dumouchel, "The spescom data voice nist sre 2004 system," in *NIST SRE 2004 Evaluation Workshop*, 2004. presented at NIST SRE 2004 Evaluation Workshop, Toledo, Spain.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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`.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[18] Available at `http://www.itl.nist.gov/iad/mig/tests/sre/2010/NIST\_SRE10\_evalplan.r6.pdf`.

[19] 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.

[20] 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.