Factorized Sub-Space Estimation for Fast and Memory Effective I-vector Extraction

Factorized Sub-space Estimation for Fast and Memory Effective I-vector Extraction

Sandro Cumani and Pietro Laface

Abstract

Most of the state–of–the–art speaker recognition systems use a compact representation of spoken utterances referred to as i–vector. Since the "standard" i–vector extraction procedure requires large memory structures and is relatively slow, new approaches have recently been proposed that are able to obtain either accurate solutions at the expense of an increase of the computational load, or fast approximate solutions, which are traded for lower memory costs. We propose a new approach particularly useful for applications that need to minimize their memory requirements. Our solution not only dramatically reduces the memory needs for i–vector extraction, but is also fast and accurate compared to recently proposed approaches. Tested on the female part of the tel-tel extended NIST 2010 evaluation trials, our approach substantially improves the performance with respect to the fastest but inaccurate eigen-decomposition approach, using much less memory than other methods.

Index Terms

Speaker Recognition, I-vectors, I-vector extraction, Probabilistic Linear Discriminant Analysis, dictionary.

I. Introduction

A simple and effective model for speaker recognition has been introduced in [1], [2]. In this approach, speaker and channel variability are modeled in a common constrained low–dimensional space spanned by the column vectors of a matrix $\mathbf{T}$, and a speech segment is represented by a low–dimensional "identity vector" or i-vector. The low dimensionality of i–vectors makes them suitable for fast classification using either generative models based on Probabilistic Linear Discriminant Analysis (PLDA) [3], [4], or discriminative classifiers such as Support Vector Machines (SVM) or Logistic Regression [5], [6], [7].

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).

Very good results have been obtained using i-vectors not only in speaker but also in several other tasks such as language recognition [8], [9], speaker segmentation [10], [11], clustering [12], [13], emotion recognition [14], [15] and age recognition [16].

Since the "standard" i–vector extraction procedure requires large memory structures and is relatively slow, new approaches have recently been proposed that are able to obtain either fast approximate solutions, [17], [18], possibly traded for lower memory costs, or accurate solutions based on a Variational Bayes (VB) formulation, at the expense of an increase of the computational load [19], [20], [21]. A simplification of the i–vector extraction is proposed in [17], based on an approximated simultaneous diagonalization of the terms composing the i–vector posterior covariance matrix. This "eigen–decomposition" approach is very fast and memory effective, but suffers a significant accuracy degradation with respect to the standard one. The approach in [18] focuses on fast approximate i–vector extraction, but it does not take care of memory issues. A VB systems saves memory because it extracts iteratively sub–blocks of i-vector elements. Performing a sufficient number of iterations, this technique is able to produce accurate i–vectors at the expense of being slower than the standard approach. In [21] we have highlighted that 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, the effectiveness of the i-vector extractor is more relevant for systems dealing with short utterances [22], [23], [24], [25] such as, for example, the text prompts in speaker verification [26], [27].

In this paper we propose a new approximate i–vector extraction approach particularly useful for applications that need to optimize their memory requirements without sensibly affecting their performance and speed. We propose a solution for the main memory cost issues in the standard i–vector extraction: the size of the variability sub–space matrix $\mathbf{T}$, and the huge amount of memory that has to be devoted for storing pre–computed matrices for the sake of computation speedup. The key idea in our solution is that it is possible to factorize the variability sub–space matrix $\mathbf{T}$ so that it is not necessary to store all its rows to perform i–vector extraction. These rows can be obtained as a linear combination of the atoms of a common dictionary. The notion of dictionary and atoms is well known in the field of sparse coding [28]. In particular, given a rank M variability sub–space, represented by a $C \times F \times M$ matrix $\mathbf{T}$, which stacks C ($F \times M$) sub-matrices $\mathbf{T}^{(c)}$, each corresponding to the c–th mixture component of a Gaussian Universal Background Model (UBM) with feature dimension F, and C components, we demonstrate that

a good approximation of the $\mathbf{T}^{(c)}$ matrices can be obtained by means of the decomposition:

$$\hat{\mathbf{T}}^{(c)} = \mathbf{O}^{(c)}\mathbf{\Pi}^{(c)}\mathbf{Q} \approx \mathbf{T}^{(c)} \ , \tag{1}$$

where $\mathbf{O}^{(c)}$ is an orthogonal $F \times F$ matrix, $\mathbf{\Pi}^{(c)}$ is a sparse $F \times K$ matrix having at most one non-zero element per row, and $\mathbf{Q}$ is a $K \times M$ dictionary matrix, shared among all $\hat{\mathbf{T}}^{(c)}$, including K atoms in its rows. $\hat{\mathbf{T}}^{(c)}$ is, thus, a linear combination of F atoms of $\mathbf{Q}$. This factorization of the sub–matrices $\mathbf{T}^{(c)}$ is represented in Figure 1, which shows that each row of matrix $\mathbf{\Pi}^{(c)}$ has a single non–zero element that selects and weights a dictionary atom. The matrix of weighted atoms is then rotated and scaled by the orthogonal matrix $\mathbf{O}^{(c)}$ to finally get the approximated $\hat{\mathbf{T}}^{(c)}$.

Since the size K of the dictionary that we estimate can be selected according to memory-accuracy trade–offs, and it is usually much less than $C \times F$, our solution not only substantially improves the performance with respect to the fast, but inaccurate, eigen-decomposition approach [17], but also dramatically reduces the memory needed for i–vector extraction compared to other methods [2], [19], [21], which require storing the original sub–space matrix $\mathbf{T}$. In the experimental section, we compare the memory and computational complexity of these approaches together with their accuracy, looking for the trade–offs that make our technique suitable both for large and for small-footprint applications. Examples of such applications are not limited to speaker authentication in smartphones or other embedded systems, because memory is a precious resource even for applications running on servers. Another advantage of saving memory is that it is also possible to use larger and possibly more precise models if more data become available.

The paper is organized as follows: Section II summarizes the i–vector representation for speaker recognition, setting the background for i–vector computation. Section III recalls two recently proposed memory–aware i–vector extraction techniques, the eigen–decomposition and the Variational Bayes approaches, and analyzes their computational and memory complexity in order to give motivations for our factorized sub–space approach, which is illustrated in Section IV. Section V is devoted to the estimation of the matrices that are necessary to factorize the matrices $\mathbf{T}^{(c)}$, and in Section VI we detail the steps for obtaining full-rank approximated $\hat{\mathbf{T}}^{(c)}$ matrices. I–vector extraction by means of a Conjugate Gradient procedure is illustrated in Section VII, and its complexity is analyzed in Section VIII. The experimental results are presented and commented in Section IX, and conclusions are drawn in Section X.

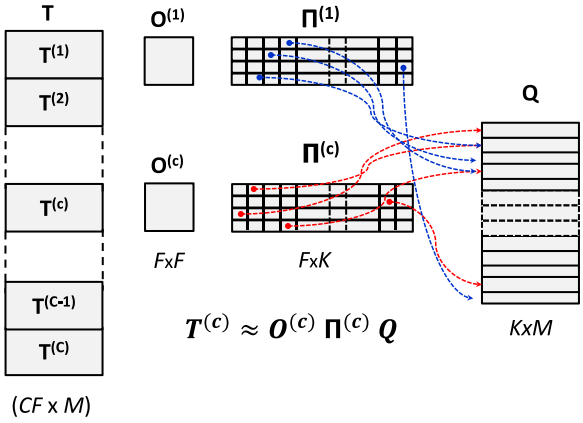

Fig. 1: Factorization of the sub–matrices $\mathbf{T}^{(c)}$. C is the number of mixture components of a Gaussian Universal Background Model with feature dimension F. M is the dimension of the variability sub–space, $\mathbf{O}^{(c)}$ is an orthogonal matrix, $\mathbf{\Pi}^{(c)}$ is a sparse matrix having at most one non-zero element per row, and $\mathbf{Q}$ is a dictionary matrix, shared among all the sub–matrices $\mathbf{T}^{(c)}$, with K atoms in its rows.

II. I–VECTOR REPRESENTATION

The i–vector representation [1], [2] constrains the GMM supervector $\mathbf{s}$, representing both the speaker and channel characteristics of a given speech segment, to live in a single sub–space according to:

$$\mathbf{s} = \mathbf{m} + \mathbf{\Sigma}^{\frac{1}{2}}\mathbf{T}\mathbf{w} \ , \tag{2}$$

where $\mathbf{m}$ is the UBM supervector, $\mathbf{T}$ is a low-rank rectangular matrix with $C \times F$ rows and M columns. The M columns of $\mathbf{T}$ are vectors spanning the variability space, and $\mathbf{w}$ is a random vector of size M having a standard normal prior distribution. $\mathbf{T}$ is multiplied for convenience by $\mathbf{\Sigma}^{\frac{1}{2}}$, where $\mathbf{\Sigma}$ denotes the block–diagonal matrix whose diagonal blocks contain the UBM covariance matrices $\mathbf{\Sigma}^{(c)}$. It is worth noting that the i–vector representation (2) is equivalent to the classical one, but takes advantage of the UBM statistics whitening introduced in [17] to simplify the i–vector extraction.

Given a set of feature vectors $\mathcal{X} = \{\mathbf{x}_1, \dots \mathbf{x}_t \dots \mathbf{x}_T\}$ extracted for a speech segment, the corresponding i–vector $\mathbf{w}_\mathcal{X}$ is computed as the mean of the posterior distribution $\mathrm{p}(\mathbf{w}|\mathcal{X})$:

$$\mathbf{w}_\mathcal{X} = \mathbf{L}_\mathcal{X}^{-1}\mathbf{T}^{*}\mathbf{f}_\mathcal{X} \ , \tag{3}$$

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

$$\mathbf{L}_\mathcal{X} = \mathbf{I} + \sum_c N_\mathcal{X}^{(c)}\mathbf{T}^{(c)*}\mathbf{T}^{(c)} \ . \tag{4}$$

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 in $\mathcal{X}$, $\mathbf{T}^{(c)}$ 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 covariance–normalized first–order statistics $\mathbf{f}_{\mathcal{X}}^{(c)}$, centered around the corresponding UBM means:

$$\mathbf{f}_{\mathcal{X}}^{(c)} = \mathbf{\Sigma}^{(c)^{-\frac{1}{2}}} \left[\sum_t \left(\gamma_t^{(c)} \mathbf{x}_t \right) - N_{\mathcal{X}}^{(c)} \mathbf{m}^{(c)} \right] , \tag{5}$$

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

III. Memory aware i–vector extraction

The complexity of a single i–vector computation (3) 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 (3) plus $O(CFM^2)$ for (4). Usually the number of Gaussian components C is greater than the sub–space 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 $O(CFM)$. In Section IX, devoted to the experiments, we refer to this approach as the "slow baseline".

The standard solution for (4) is obtained, however, by pre–computing and storing for each mixture component c its covariance matrix $\mathbf{T}^{(c)*}\mathbf{T}^{(c)}$. The computational cost is reduced to $O(CM^2)$, but this speed–up comes at the expense of an additional $O(CM^2)$ memory demand for computing (4), which dominates the other memory costs.

Among the approaches that have been recently proposed to cope both with memory constraints and computational load, we will consider as benchmarks in this work the fast, but inaccurate "eigen–decomposition" approach, and the accurate i–vector extraction methods based on a Variational Bayes (VB) formulation.

In the eigen–decomposition approach [17], a simultaneous approximate diagonalization of the matrices $\mathbf{T}^{(c)*}\mathbf{T}^{(c)}$ has been introduced for fast computation of the i–vectors with low memory resources. In this approach, each $\mathbf{T}^{(c)*}\mathbf{T}^{(c)}$ is approximated by a diagonal matrix $\bar{\mathbf{D}}^{(c)}$. This approximation has the remarkable advantage that a diagonal $\bar{\mathbf{L}}_{\mathcal{X}}$ is obtained, which can be computed by C element–wise products of two vectors of dimension M, and its inversion cost becomes negligible. 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 (3). This cost dominates because computing the diagonal matrix $\bar{\mathbf{L}}_{\mathcal{X}}$ has complexity $O(CM)$, and its inversion is

just $O(M)$. The contribution $O(M^2)$ for the back–rotation of $\bar{\mathbf{L}}_{\mathcal{X}}$ to obtain the approximate i–vector 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 computing $\bar{\mathbf{L}}_{\mathcal{X}}$ and storing the back–rotation matrices, respectively. These additional costs, however, are relatively small because $CF \gg M$.

This approach is very fast and memory effective, and its performance as reported in [17] is good, but it does not reach the accuracy of the standard approach. Thus, alternative memory-aware accurate i–vector extraction methods have been recently introduced, based on a Variational Bayes (VB) formulation [19], [20], [21]. In this framework, an i-vector is obtained by iterating the estimation of one sub-block of ivector elements at a time, keeping fixed all the others. It is worth noting that the VB approach computes i–vectors as accurate as the ones obtained by the standard technique, but requires only a fraction of its memory, the same memory required by the slow baseline approach to keep in memory matrix $\mathbf{T}$, i.e., $O(CFM)$. This technique is, however, slower than the standard one.

We present in the next sections a new approximate i–vector extraction approach that requires much less memory than the other mentioned techniques, and gives higher performance compared to the eigen–decomposition by using comparable or even less processing resources. In Section IV we present a decomposition of the sub–matrices $\mathbf{T}^{(c)}$ of $\mathbf{T}$ that allows $\mathbf{T}$ to be compressed by using a dictionary shared among all the sub–matrices $\mathbf{T}^{(c)}$. In Section VII we show that using this decomposition together with a Conjugate Gradient procedure it is possible to avoid the computation, and inversion, of the precision matrix $\mathbf{L}_{\mathcal{X}}$. We also show that the i–vector extraction is memory effective because it does not need storing the pre–computed covariance matrices $\mathbf{T}^{(c)*}\mathbf{T}^{(c)}$, and is fast because it performs, iteratively, simple diagonal and matrix–vector products of matrices smaller than the ones used by the other approaches.

IV. Factorized sub–space estimation of matrix $\mathbf{T}$

Let's consider the decomposition of sub–matrix $\mathbf{T}^{(c)}$ of $\mathbf{T}$:

$$\mathbf{T}^{(c)} = \mathbf{O}^{(c)}\mathbf{\Pi}_M^{(c)}\mathbf{G}^{(c)*} \ , \tag{6}$$

where $\mathbf{O}^{(c)}$ is an orthogonal $F \times F$ matrix, $\mathbf{\Pi}_M^{(c)}$ is an $F \times M$ matrix having at most one non–null element per row, and $\mathbf{G}^{(c)}$ is a $M \times M$ orthogonal matrix. The decomposition is not unique, but its existence is guaranteed by the existence of the Singular Value Decomposition (SVD) of $\mathbf{T}^{(c)}$, which can be considered as a particular case of (6) where $\mathbf{\Pi}_M^{(c)}$ has non–negative entries on its diagonal. The decomposition (6) can be rewritten in a compact form by stacking all matrices $\mathbf{G}^{(c)}$ in a single matrix

$\mathbf{G}$:

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}^{(1)*} \\ \vdots \\ \mathbf{G}^{(C)*} \end{bmatrix} , \tag{7}$$

and by replacing each matrix $\mathbf{\Pi}_M^{(c)}$ by a $F \times (CM)$ matrix $\mathbf{\Pi}_{CM}^{(c)}$ obtained by appropriately zero–padding each $\mathbf{\Pi}_M^{(c)}$ as:

$$\mathbf{\Pi}_{CM}^{(c)} = \left[\underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{c-1}, \mathbf{\Pi}_M^{(c)}, \underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{C-c} \right] . \tag{8}$$

Matrix $\mathbf{T}^{(c)}$ can then be computed as:

$$\mathbf{T}^{(c)} = \mathbf{O}^{(c)} \mathbf{\Pi}_{CM}^{(c)} \mathbf{G} . \tag{9}$$

It is worth noting that the c-th matrix $\mathbf{\Pi}_{CM}^{(c)}$ acts as a selector of F rows of $\mathbf{G}$, thus (9) can be simplified, keeping only the rows of $\mathbf{G}$ which are selected by at least one $\mathbf{\Pi}_{CM}^{(c)}$ matrix, and resizing accordingly the matrices $\mathbf{\Pi}_{CM}^{(c)}$, obtaining:

$$\mathbf{T}^{(c)} = \mathbf{O}^{(c)} \mathbf{\Pi}_{CF}^{(c)} \bar{\mathbf{G}} , \tag{10}$$

where $\bar{\mathbf{G}}$ is a $(CF) \times M$ matrix, and $\mathbf{\Pi}_{CF}^{(c)}$ has dimension $F \times (CF)$. The i–vector posterior precision matrix can be, thus, computed as:

$$\sum_{c=1}^{C} N_{\mathcal{X}}^{(c)} \mathbf{T}^{(c)*} \mathbf{T}^{(c)} + \mathbf{I} = \mathbf{G}^* \left(\sum_{c=1}^{C} N_{\mathcal{X}}^{(c)} \mathbf{\Pi}_{CF}^{(c)*} \mathbf{\Pi}_{CF}^{(c)} \right) \mathbf{G} + \mathbf{I} , \tag{11}$$

where each product $\mathbf{\Pi}_{CF}^{(c)*} \mathbf{\Pi}_{CF}^{(c)}$ is a diagonal matrix with F non-zero elements. The computational complexity of (11) and (4) is the same, however, it is possible to get an accurate approximation $\hat{\mathbf{T}}^{(c)}$ of each matrix $\mathbf{T}^{(c)}$ by replacing matrix $\mathbf{G}$ by a smaller $K \times M$ matrix $\mathbf{Q}$ as:

$$\hat{\mathbf{T}}^{(c)} = \mathbf{O}^{(c)} \mathbf{\Pi}^{(c)} \mathbf{Q} \approx \mathbf{T}^{(c)} , \tag{12}$$

where $\mathbf{\Pi}^{(c)}$ is a sparse $F \times K$ matrix with at most one non–zero element per row. Each matrix $\hat{\mathbf{T}}^{(c)}$ is thus obtained by a linear combination of F vectors, selected from a set of K atoms of a shared dictionary represented by the $K \times M$ matrix $\mathbf{Q}$. The original $\mathbf{T}$ matrix could be recovered selecting $K = CF$, but we are able to greatly reduce the memory needs for storing matrix $\mathbf{T}$, and for computing the i–vector posterior, by selecting a small value for K. In Section IX we show that small values of K are sufficient to get good accuracy.

V. MATRIX $\mathbf{T}^{(c)}$ APPROXIMATION

The matrices $\mathbf{O}^{(c)}$, $\mathbf{\Pi}^{(c)}$, and $\mathbf{Q}$ are obtained by minimizing a weighted average square norm of the difference between each $\mathbf{T}^{(c)}$ and its approximation $\hat{\mathbf{T}}^{(c)}$. In particular, if $\omega^{(c)}$ is the weight of the c–th component of the UBM, the objective function that we optimize is:

$$\min_{\{\mathbf{O}^{(c)}\},\{\mathbf{\Pi}^{(c)}\},\mathbf{Q}} \sum_c \omega^{(c)} \left\| \mathbf{T}^{(c)} - \mathbf{O}^{(c)}\mathbf{\Pi}^{(c)}\mathbf{Q} \right\|^2 . \tag{13}$$

where matrices $\mathbf{O}^{(c)}$ are constrained to be orthogonal, and matrices $\mathbf{\Pi}^{(c)}$ are constrained to have at most one non–zero element per row.

In the following, all optimizations with respect to $\mathbf{O}^{(c)}$ and $\mathbf{\Pi}^{(c)}$ are assumed to have the same constraints even if not explicitly mentioned.

The optimization is performed by updating a matrix while keeping constant the others, according to the iterative sequence of optimizations illustrated in Algorithm 1. In our experiments, 10 iterations of alternate optimizations of $\mathbf{\Pi}^{(c)}$ and $\mathbf{O}^{(c)}$ are performed before a new matrix $\mathbf{Q}$ is estimated keeping fixed $\mathbf{\Pi}^{(c)}$ and $\mathbf{O}^{(c)}$. This procedure is repeated for 40 iterations. These empirical settings are conservative because the iterative optimization is fast and done in training once for all. The optimization solutions for the three terms of the factorized sub–space decomposition are presented in the next sub–sections beginning with $\mathbf{Q}$ and $\mathbf{O}^{(c)}$, because they are easier to derive. Then, we illustrate the optimization with respect to $\mathbf{\Pi}^{(c)}$, which is done in two steps in order to obtain a full–rank matrix.

In order to derive the update equations for the factorized sub–space decomposition matrices we rewrite our objective function (13) as:

$$\begin{aligned} \min_{\{\mathbf{O}^{(c)}\},\{\mathbf{\Pi}^{(c)}\},\mathbf{Q}} \sum_c \omega^{(c)} \Big[& \operatorname{tr}\left(\mathbf{T}^{(c)*}\mathbf{T}^{(c)}\right) + \\ & \operatorname{tr}\left(\mathbf{Q}^{*}\mathbf{D}^{(c)}\mathbf{Q}\right) - 2\operatorname{tr}\left(\mathbf{T}^{(c)*}\mathbf{O}^{(c)}\mathbf{\Pi}^{(c)}\mathbf{Q}\right)\Big] , \end{aligned} \tag{14}$$

where $\mathbf{D}^{(c)} = \mathbf{\Pi}^{(c)*}\mathbf{\Pi}^{(c)}$ is a diagonal matrix.

A. Matrix $\mathbf{Q}$ optimization

We solve for $\mathbf{Q}$ by zeroing the gradient of (14) with respect to Q :

$$\sum_c \left(2\,\omega^{(c)}\mathbf{D}^{(c)}\mathbf{Q} - 2\,\omega^{(c)}\mathbf{\Pi}^{(c)*}\mathbf{O}^{(c)*}\mathbf{T}^{(c)} \right) \tag{15}$$

Algorithm 1
1. Initialize the matrices $\mathbf{Q}$ and $\mathbf{O}^{(c)}$ (Section V-D).
2. **for** $i \leftarrow 1$ **to** $num\_outer\_iterations$ **do**
3. **for** $j \leftarrow 1$ **to** $num\_inner\_iterations$ **do**
4. Update $\mathbf{\Pi}^{(c)}$ by equation (31) keeping constant $\mathbf{Q}$ and $\mathbf{O}^{(c)}$
5. Update $\mathbf{O}^{(c)}$ by equation (22) keeping constant $\mathbf{Q}$ and $\mathbf{\Pi}^{(c)}$
6. Update $\mathbf{Q}$ by equations (16) and (17) keeping constant $\mathbf{O}^{(c)}$ and $\mathbf{\Pi}^{(c)}$

while keeping fixed all matrices $\mathbf{O}^{(c)}$ and $\mathbf{\Pi}^{(c)}$, obtaining:

$$\mathbf{Q} = \left(\sum_c \omega^{(c)} \mathbf{D}^{(c)} \right)^{-1} \left(\sum_c \omega^{(c)} \mathbf{\Pi}^{(c)*} \mathbf{O}^{(c)*} \mathbf{T}^{(c)} \right) \ . \tag{16}$$

The pseudo–inverse of the diagonal matrix $\sum_c \omega^{(c)} \mathbf{D}^{(c)}$ is computed to take care of its possible singularities, i.e., leaving unchanged the tiny or zero diagonal values.

It is worth noting that our training procedure cannot guarantee that matrix $\mathbf{Q}$ is full rank, but we could make it full rank by appending to the actual dictionary matrix $\mathbf{Q}$ a (full rank) $M \times M$ identity matrix whose rows are never re–estimated. However, we observed that in practice no particular care has to be taken to avoid rank deficient $\mathbf{Q}$ matrices even for small sized dictionaries.

Without loss of generality, after each iteration that estimates a new matrix $\mathbf{Q}$, we normalize its rows, and we update accordingly the corresponding entries in the $\mathbf{\Pi}^{(c)}$ matrices. In particular, defining $\mathbf{\Delta} = \mathbf{Q}\mathbf{Q}^* \circ \mathbf{I}$, where $\circ$ is the element–wise matrix product, the matrices $\mathbf{Q}$ and $\mathbf{\Pi}^{(c)}$ are updated as:

$$\mathbf{Q} \leftarrow \mathbf{\Delta}^{-1} \mathbf{Q} \ , \qquad \mathbf{\Pi}^{(c)} \leftarrow \mathbf{\Pi}^{(c)} \mathbf{\Delta} \ , \tag{17}$$

so that the objective function is not affected because the normalization factor is taken into account by matrix $\mathbf{\Pi}^{(c)}$. This procedure allows us avoiding that large values in $\mathbf{Q}$ lead to small estimated $\mathbf{\Pi}^{(c)}$ values. This bad behavior could reinforce itself in successive iterations.

B. Matrix $\mathbf{O}^{(c)}$ optimization

Since the optimization of each matrix $\mathbf{O}^{(c)}$ can be done independently from the others, we can maximize the third term in (14) keeping constant the matrices $\mathbf{\Pi}^{(c)}$ and $\mathbf{Q}$ as:

$$\max_{\mathbf{O}^{(c)}} \ \mathrm{tr} \left(\mathbf{T}^{(c)*} \mathbf{O}^{(c)} \mathbf{\Pi}^{(c)} \mathbf{Q} \right) \ . \tag{18}$$

Since the trace operator is invariant under cyclic permutations, i.e., $\mathrm{tr}(\mathbf{ABC}) = \mathrm{tr}(\mathbf{CAB})$, (18) can be rewritten as:

$$\max_{\mathbf{O}^{(c)}} \mathrm{tr}\left(\mathbf{O}^{(c)}\mathbf{\Pi}^{(c)}\mathbf{Q}\mathbf{T}^{(c)*}\right) = \max_{\mathbf{O}^{(c)}} \mathrm{tr}\left(\mathbf{O}^{(c)}\mathbf{Z}\right) \ , \tag{19}$$

where $\mathbf{Z} = \mathbf{\Pi}^{(c)}\mathbf{Q}\mathbf{T}^{(c)*}$. Since we solve for a single component c at a time, for the sake of readability of the equations in this section, the superscript (c) in $\mathbf{Z}$ has been dropped.

The Von Neumann's trace inequality [29], [30] states that:

$$\left|\mathrm{tr}(\mathbf{O}^{(c)}\mathbf{Z})\right| \leq \sum_{i=1}^{F} \sigma_{oi}\sigma_{zi} \ , \tag{20}$$

where σ_{oi} and σ_{zi} are the sorted i–th singular values obtained by SVD of $\mathbf{O}^{(c)}$ and $\mathbf{Z}$, respectively. Since $\mathbf{O}^{(c)}$ has to be orthogonal, its singular values must be equal to 1. Thus, for any feasible solution $\mathbf{O}^{(c)}$, the objective function is bounded by:

$$\mathrm{tr}(\mathbf{O}^{(c)}\mathbf{Z}) \leq \sum_{i=1}^{F} \sigma_{zi} \ , \tag{21}$$

which is maximized if we find a matrix $\mathbf{O}^{(c)}$ such that the singular values of $\mathbf{O}^{(c)}\mathbf{Z}$ and $\mathbf{Z}$ are exactly the same. This condition is satisfied by matrix:

$$\mathbf{O}^{(c)} = \mathbf{V}_{\mathbf{Z}}\mathbf{U}_{\mathbf{Z}}^{*} \ , \tag{22}$$

where $\mathbf{V}_{\mathbf{Z}}$ and $\mathbf{U}_{\mathbf{Z}}$ are the singular vectors of the SVD of $\mathbf{Z} = \mathbf{\Pi}^{(c)}\mathbf{Q}\mathbf{T}^{(c)*}$, decomposed as $\mathbf{Z} = \mathbf{U}_{\mathbf{Z}}\mathbf{\Sigma}_{\mathbf{Z}}\mathbf{V}_{\mathbf{Z}}^{*}$. This can be verified substituting (22) in the left hand side of (21).

C. Matrix $\mathbf{\Pi}^{(c)}$ optimization

Considering again the last two terms of (14), the optimization can be done independently for each $\mathbf{\Pi}^{(c)}$, considering constants $\mathbf{O}^{(c)}$ and $\mathbf{Q}$.

Using the permutation property of the trace for the second term we get:

$$\mathrm{tr}\left(\mathbf{Q}^{*}\mathbf{\Pi}^{(c)*}\mathbf{\Pi}^{(c)}\mathbf{Q}\right) = \mathrm{tr}\left(\mathbf{\Pi}^{(c)*}\mathbf{\Pi}^{(c)}\mathbf{Q}\mathbf{Q}^{*}\right) \ . \tag{23}$$

Although the dimension of $\mathbf{QQ}^{*}$ is huge, we need only its diagonal because, for any feasible solution $\mathbf{\Pi}^{(c)*}$, matrix $\mathbf{\Pi}^{(c)*}\mathbf{\Pi}^{(c)}$ is diagonal. Moreover, since the atoms of the dictionary matrix $\mathbf{Q}$ are normalized, the diagonal elements of $\mathbf{QQ}^{*}$ are $q_k = 1$. Thus, the second term of the objective function (14) can be

written as:

$$\begin{aligned}\mathrm{tr}\left(\mathbf{\Pi}^{(c)*}\mathbf{\Pi}^{(c)}\mathbf{Q}\mathbf{Q}^{*}\right) &= \sum_{k=1}^{K} q_k^2 \left(\sum_{f=1}^{F} {\boldsymbol{\pi}_f^{(c)}}^{*} \boldsymbol{\pi}_f^{(c)}\right)_{k,k} \\ &= \sum_{k=1}^{K}\sum_{f=1}^{F} ({\boldsymbol{\pi}_f^{(c)}}^{*} \boldsymbol{\pi}_f^{(c)})_{k,k} \ ,\end{aligned} \tag{24}$$

where $\boldsymbol{\pi}_f^{(c)}$ is the f-th row of $\mathbf{\Pi}^{(c)}$.

Applying again the invariance rule for cyclic permutations of the trace operator, and recalling that the trace of a matrix is equal to the trace of its transpose, the third term of (14) can be rewritten as:

$$\mathrm{tr}\left(\mathbf{T}^{(c)*}\mathbf{O}^{(c)}\mathbf{\Pi}^{(c)}\mathbf{Q}\right) = \mathrm{tr}\left(\mathbf{\Pi}^{(c)*}\mathbf{O}^{(c)*}\mathbf{T}^{(c)}\mathbf{Q}^{*}\right) \ . \tag{25}$$

Defining

$$\mathbf{A}^{(c)} = \mathbf{O}^{(c)*}\mathbf{T}^{(c)}\mathbf{Q}^{*} \ , \tag{26}$$

$\mathbf{\Pi}^{(c)}$ can be obtained optimizing:

$$\begin{aligned}&\sum_{k=1}^{K}\sum_{f=1}^{F} (\boldsymbol{\pi}_f^{(c)*}\boldsymbol{\pi}_f^{(c)})_{k,k} - 2\sum_{f=1}^{F} \mathrm{tr}\left(\boldsymbol{\pi}_f^{(c)*} A_f^{(c)}\right) = \\ &\sum_{f=1}^{F}\left[\sum_{k=1}^{K} (\boldsymbol{\pi}_f^{(c)*}\boldsymbol{\pi}_f^{(c)})_{k,k} - 2\,\mathrm{tr}\left(\boldsymbol{\pi}_f^{(c)*} A_f^{(c)}\right)\right] \ ,\end{aligned} \tag{27}$$

where $A_f^{(c)}$ is the f–th row of $A^{(c)}$. We can optimize each $\boldsymbol{\pi}_f^{(c)}$ independently because the terms in the summation (27) can be factorized with respect to the rows $\boldsymbol{\pi}_f^{(c)}$. Thus, for a given c and f, $\boldsymbol{\pi}_f^{(c)}$ can be found by minimizing:

$$\sum_{k=1}^{K} (\boldsymbol{\pi}_f^{(c)*}\boldsymbol{\pi}_f^{(c)})_{k,k} - 2\,\mathrm{tr}\left(\boldsymbol{\pi}_f^{(c)*} A_f^{(c)}\right) \ . \tag{28}$$

Since the row vector $\boldsymbol{\pi}_f^{(c)}$ must have a single non-zero element v_k, with index k, the optimal index k_f^{opt}, and its corresponding value v_f^{opt}, are found as the solution of:

$$k_f^{\mathrm{opt}} = \underset{k}{\mathrm{argmin}} \min_{v_k} \left(v_k^2 - 2v_k A_{f,k}^{(c)}\right) \ , \tag{29}$$

i.e., for a given k, the minimum value v_k is obtained by zeroing the derivative of the function $v_k^2 - 2v_k A_{f,k}^{(c)}$:

$$v_k = A_{f,k}^{(c)} \ . \tag{30}$$

Substituting (30) in (29) we finally get:

$$k_f^{\text{opt}} = \underset{k}{\text{argmax}}\, A_{f,k}^{(c)2} \ , \qquad v_f^{\text{opt}} = A_{f,k^{\text{opt}}}^{(c)} \ . \tag{31}$$

D. Initialization of matrices $\mathbf{Q}$ *and* $\mathbf{O}^{(c)}$

Recalling that a SVD of $\mathbf{T}^{(c)}$ is a particular case of decomposition (6), in order to initialize the dictionary matrix $\mathbf{Q}$ and all $\mathbf{O}^{(c)}$ matrices, we perform the SVD of each matrix $\omega^{(c)}\mathbf{T}^{(c)}$ as:

$$\omega^{(c)}\mathbf{T}^{(c)} = \mathbf{U}^{(c)}\boldsymbol{S}^{(c)}\mathbf{V}^{(c)*} \ . \tag{32}$$

Matrix $\mathbf{O}^{(c)}$ is initialized by matrix $\mathbf{U}^{(c)}$.

Matrix $\mathbf{Q}$ is initialized by pooling together the rows of all the matrices $\mathbf{V}^{(c)*}$ and keeping only the rows corresponding to the largest pooled singular values.

Thus, we can select the size K of the dictionary $\mathbf{Q}$ to be much less than the size ($CF \times M$) of $\mathbf{T}$, according to memory-accuracy trade-offs.

No initialization is needed for the matrices $\mathbf{\Pi}^{(c)}$ because, given $\mathbf{Q}$ and $\mathbf{O}^{(c)}$, the rows of each $\mathbf{\Pi}^{(c)}$ can be set according to (31).

VI. Estimating a full–rank matrix $\mathbf{\Pi}^{(c)}$

Since the optimization of $\mathbf{\Pi}^{(c)}$ is performed independently for each row $\boldsymbol{\pi}_f^{(c)}$, it may happen that the optimal index for two rows a and b is the same, i.e., that two non–zero values appear in the same column of $\boldsymbol{\pi}_a^{(c)}$ and $\boldsymbol{\pi}_b^{(c)}$, as illustrated in Figure 2 (a). This configuration, however, indicates that matrix $\mathbf{\Pi}^{(c)}$ is not full-rank, and that two of its rows select the same dictionary atom. Thus, one of these rows is superfluous. By properly merging in a single row the information carried by both rows, tThe optimization function does not change, and additional optimization can be obtained by re–estimating, according to (31), the optimal index and value of the row that was cleared ($\boldsymbol{\pi}_b^{(c)}$ in Figure 2). In the following we present an algorithm that performs these optimizations leading to a full–rank estimate of matrix $\mathbf{\Pi}^{(c)}$, which better exploits the shared dictionary.

Let k_f denote the index of the non–zero element of $\boldsymbol{\pi}_f$, the f–th row of $\mathbf{\Pi}^{(c)}$, and let v_f be its corresponding value. Let $v_f = 0$ if all the elements of a row $\boldsymbol{\pi}_f$ are zero. Then, for matrix $\mathbf{\Pi}^{(c)}$ to be full rank it is necessary (see Proposition 1 in Appendix) that $|v_f| > 0$ for all rows f and that $k_a \neq k_b$ for $a \neq b$, i.e., $\mathbf{\Pi}^{(c)}$ must have a single non–zero element per row, located in different columns. Since the optimization procedure introduced in Section V-C is not able to directly impose such constraints on

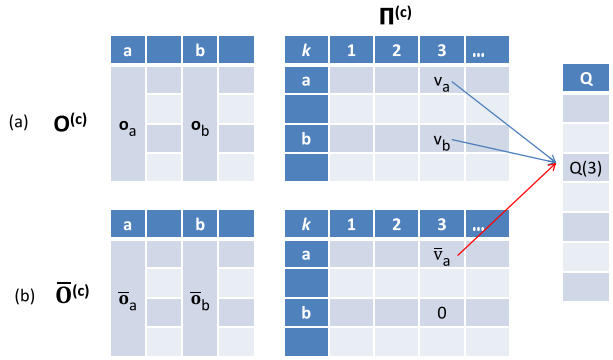

Fig. 2: Merging two rows of a matrix $\mathbf{\Pi}^{(c)}$ having two non-zero values in the same column: (a) initial configuration, (b) updates of matrix $\mathbf{\Pi}^{(c)}$ and $\mathbf{O}^{(c)}$ according to steps 2 and 3 of Algorithm 2.

matrix $\mathbf{\Pi}^{(c)}$, it may happen that the estimated matrix is not full rank. It is, therefore, possible to estimate a better matrix. Assuming that both $\mathbf{T}^{(c)}$ and $\mathbf{Q}$ are full rank, i.e., $\rho(\mathbf{T}^{(c)}) = F$, and $\rho(\mathbf{Q}) = M$, our goal is an iterative procedure that, keeping constant $\mathbf{Q}$, estimates a full–rank $\mathbf{\Pi}^{(c)}$ matrix, and decreases the objective function:

$$g(\mathbf{O}^{(c)}, \mathbf{\Pi}^{(c)}) = \left\| \mathbf{T}^{(c)} - \mathbf{O}^{(c)}\mathbf{\Pi}^{(c)}\mathbf{Q} \right\|^2 . \tag{33}$$

Let $\mathbf{\Pi}^{(c)}$ and $\mathbf{O}^{(c)}$ denote a (suboptimal) solution of our optimization problem, obtained by first optimizing with respect to $\mathbf{O}^{(c)}$, and then with respect to $\mathbf{\Pi}^{(c)}$, as described in Sections V-B and V-C, respectively. Assume also that at least two non–zero elements v_a and v_b of $\mathbf{\Pi}^{(c)}$ are on the same column $k = k_a = k_b$. We follow a two–step approach to estimate a better solution $(\bar{\mathbf{O}}^{(c)}, \hat{\mathbf{\Pi}}^{(c)})$ such that $\rho(\hat{\mathbf{\Pi}}^{(c)}) \geq \rho(\mathbf{\Pi}^{(c)})$ and $g(\bar{\mathbf{O}}^{(c)}, \hat{\mathbf{\Pi}}^{(c)}) < g(\mathbf{O}^{(c)}, \mathbf{\Pi}^{(c)})$.

First we estimate a solution $(\bar{\mathbf{O}}^{(c)}, \bar{\mathbf{\Pi}}^{(c)})$ equivalent to $(\mathbf{O}^{(c)}, \mathbf{\Pi}^{(c)})$ by properly merging rows a and b of $\mathbf{\Pi}^{(c)}$, and the corresponding columns of $\mathbf{O}^{(c)}$. We then apply (31) to get a better estimate of $\mathbf{\Pi}^{(c)}$.

Since the values v_a and v_b of matrix $\mathbf{\Pi}^{(c)}$ are on the same column, it is easy to verify that, if we replace the a–th column of $\mathbf{O}^{(c)}$ by the linear combination of the columns $\mathbf{o}_a$ and $\mathbf{o}_b$, given by $\tilde{\mathbf{o}}_a = v_a\mathbf{o}_a + v_b\mathbf{o}_b$, and the values v_a and v_b in $\mathbf{\Pi}^{(c)}$ by 1 and 0, respectively, we obtain a new solution which does not change the value of the product $\mathbf{O}^{(c)}\mathbf{\Pi}^{(c)}$ and, therefore, does not change the value of the objective function (33). This solution, however, would make matrix $\mathbf{O}^{(c)}$ no more orthogonal.

In order to recover an equivalent feasible solution, one can observe that the new solution does not depend on the values of the b–th column of $\mathbf{O}^{(c)}$ because $v_b = 0$. Moreover, it is worth noting that if a column vector $\mathbf{o}_f$ of $\mathbf{O}^{(c)}$ is divided by a given value v, and the value v_f of $\mathbf{\Pi}^{(c)}$ is multiplied by the same

Algorithm 2

1. **while** a pair of rows a and b of $\mathbf{\Pi}^{(c)}$ exists such that $k_a = k_b$ and $|v_a| > 0$, $|v_b| > 0$ **do**
2. Compute $\bar{\mathbf{O}}^{(c)}$ by replacing the columns $\mathbf{o}_a$ and $\mathbf{o}_b$ in $\mathbf{O}^{(c)}$ by $\bar{\mathbf{o}}_a$ and $\bar{\mathbf{o}}_b$, respectively (34).
3. Compute $\bar{\mathbf{\Pi}}^{(c)}$ by replacing the values v_a and v_b in $\mathbf{\Pi}^{(c)}$ by $\left(v_a^2 + v_b^2\right)^{\frac{1}{2}}$ and 0, respectively.
4. Estimate $\hat{\mathbf{\Pi}}^{(c)}$ using (31) as:

$$\hat{\mathbf{\Pi}}^{(c)} = \underset{\bar{\mathbf{\Pi}}}{\operatorname{argmin}} \left\| \mathbf{T}^{(c)} - \bar{\mathbf{O}}^{(c)} \bar{\mathbf{\Pi}} \mathbf{Q} \right\|^2 .$$

5. Update: $\mathbf{O}^{(c)} \leftarrow \bar{\mathbf{O}}^{(c)}$, $\mathbf{\Pi}^{(c)} \leftarrow \hat{\mathbf{\Pi}}^{(c)}$.

value v, the product $\mathbf{O}^{(c)}\mathbf{\Pi}^{(c)}$ does not change. Thus, the a–th column of $\mathbf{O}^{(c)}$ can be replaced by the unitary vector $\bar{\mathbf{o}}_a = \frac{v_a\mathbf{o}_a + v_b\mathbf{o}_b}{\|v_a\mathbf{o}_a + v_b\mathbf{o}_b\|}$, obtained by dividing $\tilde{\mathbf{o}}_a$ by its norm, provided that the value v_a of $\mathbf{\Pi}^{(c)}$ is multiplied by same norm. Then, v_b can be set to zero, but the column $\mathbf{o}_b$ of $\mathbf{O}^{(c)}$ has to be replaced by a unitary vector that must be orthogonal both to vector $\bar{\mathbf{o}}_a$ and to all other column vectors of $\mathbf{O}^{(c)}$. All these operations are summarized in Algorithm 2, which shows the steps for estimating a full–rank matrix $\mathbf{\Pi}^{(c)}$.

Let's define vectors:

$$\bar{\mathbf{o}}_a = \frac{(v_a\mathbf{o}_a + v_b\mathbf{o}_b)}{\left(v_a^2 + v_b^2\right)^{\frac{1}{2}}} , \qquad \bar{\mathbf{o}}_b = \frac{(v_a\mathbf{o}_b - v_b\mathbf{o}_a)}{\left(v_a^2 + v_b^2\right)^{\frac{1}{2}}} , \tag{34}$$

which are unitary and orthogonal ($\|\bar{\mathbf{o}}_a\| = 1$, $\|\bar{\mathbf{o}}_b\| = 1$ and $\bar{\mathbf{o}}_a^T\bar{\mathbf{o}}_b = 0$). Since $\bar{\mathbf{o}}_a$ and $\bar{\mathbf{o}}_b$ are linear combinations of $\mathbf{o}_a$ and $\mathbf{o}_b$, they are also orthogonal to all other columns of $\mathbf{O}^{(c)}$. Replacing the columns $\mathbf{o}_a$ and $\mathbf{o}_b$ of $\mathbf{O}^{(c)}$ by $\bar{\mathbf{o}}_a$ and $\bar{\mathbf{o}}_b$, respectively, and the values v_a and v_b of $\mathbf{\Pi}^{(c)}$ by $\bar{v}_a = \left(v_a^2 + v_b^2\right)^{\frac{1}{2}}$ and $\bar{v}_b = 0$, respectively, we get a feasible solution which does not change the value of the objective function (33). This procedure is indicated in step 2 and step 3 of Algorithm 2, and in Figure 2 (b).

It can be shown that the resulting solution is optimal with respect to all rows $\boldsymbol{\pi}_f$ of $\mathbf{\Pi}^{(c)}$, except for the null row $\boldsymbol{\pi}_b$. Since row $\boldsymbol{\pi}_b$ is null, it can be re–estimated according to (31) to increase the objective function value (steps 4 and 5).

This procedure ends in a finite number of steps, and leads to a full–rank estimate of matrix $\mathbf{\Pi}^{(c)}$. All proofs are given in Appendix.

VII. I-VECTOR EXTRACTION

Using the approximated $\hat{\mathbf{T}}^{(c)}$ of (12), the i–vector posterior precision matrix can be computed as:

$$\hat{\mathbf{L}}_{\mathcal{X}} \;=\; \mathbf{I} + \sum_c N_{\mathcal{X}}^{(c)} \mathbf{Q}^* \mathbf{\Pi}^{(c)*} \mathbf{O}^{(c)*} \mathbf{O}^{(c)} \mathbf{\Pi}^{(c)} \mathbf{Q}$$

$$= \mathbf{I} + \mathbf{Q}^* \sum_c N_{\mathcal{X}}^{(c)} \mathbf{\Pi}^{(c)*} \mathbf{\Pi}^{(c)} \mathbf{Q} \ . \tag{35}$$

It is worth recalling that each matrix $\mathbf{\Pi}^{(c)}$ is virtually large because it has dimension $F \times K$, but it has actually at most one non–zero element per row, thus it can be stored as a sparse matrix. Computing $\hat{\mathbf{L}}_{\mathcal{X}}$, however, not only requires $O(M^2)$ memory, but has complexity $O(KM^2)$. The i–vector extraction (3) can be performed, however, without actually computing and inverting matrix $\hat{\mathbf{L}}_{\mathcal{X}}$ by solving the system of linear equations:

$$\begin{aligned} \hat{\mathbf{L}}_{\mathcal{X}} \hat{\mathbf{w}}_{\mathcal{X}} &= \hat{\mathbf{T}}^* \mathbf{f}_{\mathcal{X}} \\ &= \sum_c \hat{\mathbf{T}}^{(c)*} \mathbf{f}_{\mathcal{X}}^{(c)} \\ &= \mathbf{Q}^* \sum_c \mathbf{\Pi}^{(c)*} \mathbf{O}^{(c)*} \mathbf{f}_{\mathcal{X}}^{(c)} \ . \end{aligned} \tag{36}$$

Since matrix $\hat{\mathbf{L}}_{\mathcal{X}}$ is symmetric and positive definite, the linear system of equations (36) can be solved by the Conjugate Gradient (CG) method [31], which solves $\hat{\mathbf{L}}_{\mathcal{X}} \hat{\mathbf{w}}_{\mathcal{X}} = \mathbf{c}$, where $\mathbf{c}$ is the right–hand side of (36), iterating from an initial guess $\hat{\mathbf{w}}_0$, and generating successive vectors that are closer to the solution $\hat{\mathbf{w}}_{\mathcal{X}}$ that minimizes the quadratic function:

$$f(\hat{\mathbf{w}}_{\mathcal{X}}) = \frac{1}{2} \hat{\mathbf{w}}_{\mathcal{X}}^* \hat{\mathbf{L}}_{\mathcal{X}} \hat{\mathbf{w}}_{\mathcal{X}} - \hat{\mathbf{w}}_{\mathcal{X}}^* \mathbf{c} \ . \tag{37}$$

The iteration updates in the CG algorithm are based on the product of matrix $\hat{\mathbf{L}}_{\mathcal{X}}$ by a vector, which is either $\hat{\mathbf{w}}_n$ or a direction vector $\mathbf{p}_n$, where n is the iteration number. Exploiting these characteristics of the algorithm, it is possible to reduce the high memory demands and the costs due to the computation and inversion of matrix $\hat{\mathbf{L}}_{\mathcal{X}}$ because the latter always appears multiplied by $\hat{\mathbf{w}}_n$ or $\mathbf{p}_n$.

The product of $\hat{\mathbf{L}}_{\mathcal{X}}$ by a generic M–dimensional vector $\mathbf{v}_n$:

$$\hat{\mathbf{L}}_{\mathcal{X}} \mathbf{v}_n = \mathbf{I} \mathbf{v}_n + \mathbf{Q}^* \left(\sum_c N_{\mathcal{X}}^{(c)} \mathbf{\Pi}^{(c)*} \mathbf{\Pi}^{(c)} \right) \mathbf{Q} \mathbf{v}_n \tag{38}$$

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

$$\begin{aligned} \mathbf{z} &= \mathbf{Q} \mathbf{v}_n \\ \mathbf{z} &\leftarrow \left(\sum_c N_{\mathcal{X}}^{(c)} \mathbf{\Pi}^{(c)*} \mathbf{\Pi}^{(c)} \right) \mathbf{z} \\ \mathbf{z} &\leftarrow \mathbf{Q}^* \mathbf{z} \\ \hat{\mathbf{L}}_{\mathcal{X}} \mathbf{v}_n &= \mathbf{z} + \mathbf{v}_n \ . \end{aligned} \tag{39}$$

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 $\sum_c N_{\mathcal{X}}^{(c)} \mathbf{\Pi}^{(c)*} \mathbf{\Pi}^{(c)}$, and finally $\hat{\mathbf{L}}_{\mathcal{X}} \mathbf{v}_n$ is obtained by the last two operations.

Pre–conditioning the conjugate gradient method, by multiplying the residual by a fixed symmetric positive-definite matrix, speeds–up convergence. The pre–condition matrix that has been used in our experiments is the inverse of:

$$\mathbf{M} = \left(\sum_c N_{\mathcal{X}}^{(c)} \right) \sum_c \omega^{(c)} \operatorname{diag}(\mathbf{T}^{(c)*} \mathbf{T}^{(c)}) + \mathbf{I} \ , \tag{40}$$

i.e., the inverse of a diagonal approximation of the precision matrix of the posterior distribution (4).

TABLE I: Results for the extended NIST SRE2010 female tests in terms of % EER, minDCF08$\times$1000 and minDCF10$\times$1000 using different i–vector extraction approaches. Label VB b–t refers to the Variational Bayes approach of [21] setting the sub–block dimension to b, and the stopping threshold to t. Label FSE–K–t refers to the Factorized Sub–space Estimation approach setting the dictionary dimension to K, and the stopping threshold to t.

System	Memory (MB)	1 core **1000 utterances**		12 cores **5000 utterances**		PLDA		
		cpu time	time ratio	cpu time	time ratio	(%) EER	min DCF08	min DCF10
Fast baseline	815	174	4.70	184	8.76	3.59	181	566
Slow baseline	188	2462	66.54	2020	96.19	3.59	181	566
VB10-10	205	385	10.4	267	12.71	3.57	181	570
VB20-10	221	338	9.14	249	11.86	3.51	182	569
VB10-100	205	228	6.16	147	7.00	3.59	184	587
VB20-100	221	195	5.27	147	7.00	3.53	183	572
Eigen–decomposition	191	37	**1.00**	21	**1.00**	4.27	201	692
FSE-2k-10 No Prec.	32.1	28	0.76	17	0.81	3.70	191	575
FSE-2k-100 No Prec.	32.1	21	0.57	15	0.71	4.04	194	581
FSE-2k-10 Diag Prec.	32.1	29	0.78	21	1.00	3.73	191	579
FSE-2k-100 Diag Prec.	32.1	24	0.65	19	0.90	3.86	200	594
FSE-3.5k-10 No Prec.	35.1	36	0.97	22	1.05	3.76	195	551
FSE-3.5k-100 No Prec.	35.1	25	0.68	18	0.86	4.08	201	588
FSE-3.5k-10 Diag Prec.	35.1	34	0.92	25	1.19	3.73	196	545
FSE-3.5k-100 Diag Prec.	35.1	28	0.76	22	1.05	3.91	200	572
FSE-5k-10 No Prec.	38	43	1.16	28	1.33	3.49	185	580
FSE-5k-100 No Prec.	38	29	0.78	22	1.05	3.69	191	606
FSE-5k-10 Diag Prec.	38	40	1.08	30	1.43	3.43	185	582
FSE-5k-100 Diag Prec.	38	31	0.84	26	1.24	3.61	188	605
FSE-10k-10 No Prec.	48	85	2.30	50	2.38	3.56	185	584
FSE-10k-100 No Prec.	48	53	1.43	35	1.67	3.76	190	589
FSE-10k-10 Diag Prec.	48	68	1.84	46	2.19	3.56	184	578
FSE-10k-100 Diag Prec.	48	49	1.32	36	1.71	3.73	190	599

TABLE II: PLDA results for the FSE approach using very small dictionary dimensions: $K = 500$ and $K = 1000$.

System	Memory (MB)	% EER	min DCF08	min DCF10
FSE-0.5k-10	29.8	5.05	247	698
FSE-0.5k-100	29.8	5.23	255	695
FSE-1k-10	30.6	4.43	227	641
FSE-1k-100	30.6	4.72	229	635

VIII. Complexity analysis

The standard, eigen-decomposition, and the VB approaches cannot avoid the cost of storing matrix $\mathbf{T}$, which is $O(CFM)$. In our FSE approach, instead, memory demand depends on K, the number of atoms in the dictionary $\mathbf{Q}$. In particular, the total memory cost is $O(KM)$ for $\mathbf{Q}$, $O(CF^2)$ for the $\mathbf{O}^{(c)}$ matrices, and $O(CF)$ for the sparse matrices $\mathbf{\Pi}^{(c)}$. Additional $O(M)$ memory has to be allocated for the pre–conditioning matrix (40) that allows reducing the number of Conjugate Gradient iterations. After training has been completed, further memory saving can be obtained by selecting the entries in each $\mathbf{\Pi}^{(c)}$ having a small absolute value, and clearing the corresponding columns of $\mathbf{O}^{(c)}$. These columns would give a negligible or no contribution to $\hat{\mathbf{T}}^{(c)} = \mathbf{O}^{(c)}\mathbf{\Pi}^{(c)}\mathbf{Q}$ because the corresponding rows of $\mathbf{\Pi}^{(c)}\mathbf{Q}$ are approximately zero. Also, the dictionary can be compacted excluding the atoms that are never used (about 200 in our FSE-5k models).

As far as the computation complexity of the FSE method is concerned, the first and the third operation in (39) have complexity $O(NKM)$, where N is the number of CG iterations. The complexity of the second operation is $O(NCF)$ because each matrix $\mathbf{\Pi}^{(c)}$ includes F entries only, plus a minor contribution $O(NK)$ for scaling $\mathbf{z}$. The complexity of the last operation is negligible because it is $O(NM)$ only. Computing $\mathbf{c}$ has complexity $O(CF^2 + KM)$.

Since few iterations (less than 10) are usually necessary to obtain approximate i–vectors that produce very good results, the Conjugate Gradient approach, having an overall complexity $O(NKM + CF^2)$ is not only faster than the standard approach, which is $O(CFM^2)$, but also of the $O(CFM)$ eigen–decomposition approach. As shown in Table I, for large values of K the FSE method becomes slower than the eigen–decomposition approach, which, however, uses far more memory, and is less accurate.

IX. Experimental settings and results

In this work we followed the same experimental protocol and settings that were presented in [20]. Since these works were focused on memory and computational costs of the i–vector extraction module, we did not devote particular care to select the best combination of features, techniques, and training data that allow obtaining the best performance. Thus, we tested our systems mainly on the female part of tel-tel extended NIST 2010 evaluation trials [32]. We tested the "standard", the Variational Bayes, the eigen–decomposition, and the FSE i–vector extraction techniques, with 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-components GMM, and 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.

Our classifier for these experiments is based on Gaussian PLDA, implemented according to the framework illustrated in [12]. The scores presented are not normalized. We trained models with full–rank channel factors, using 120 dimensions for the speaker factors. The i–vectors of the PLDA models are L_2 normalized after Within Class Covariance Normalization (WCCN) [33] has been applied. The WCCN transformations, and the PLDA models have been trained using the previously mentioned NIST datasets, and additionally the Switchboard II, Phases 2 and 3, and Switchboard Cellular, Parts 1 and 2 datasets.

Table I summarizes the performance of the evaluated approaches on the female part of the extended telephone condition in the NIST 2010 evaluation. In this table, and in the following we will refer to the standard approach as the "fast baseline" approach. The recognition accuracy is given in terms of percent Equal Error Rate (EER) and Minimum Detection Cost Functions ($\times 1000$) defined by NIST for the 2008 (minDCF08) and 2010 (minDCF10) evaluations [32].

As far as the computational speedup is concerned, the results of the experiments reported cannot be directly compared with the ones given in [21], [20] because a 5 times larger number of conversation segments have been processed in order to obtain more accurate measurements for the faster techniques. In particular, the computation time has been evaluated for the extraction of the i–vectors of 1000 and 5000 segments for the single–thread and multi–thread setting, respectively. Moreover, the absolute

TABLE III: Results for all conditions of the extended NIST SRE2010 male tests in terms of % EER, minDCF08$\times$1000 and minDCF10$\times$1000 using the standard and the Factorized Sub–space Estimation approach with dictionary dimension $K = 5000$. Condition labels Nve, Hve, and Lve refer to Normal, High, and Low vocal effort, respectively.

System		Standard			FSE		
Condition	Number	% EER	min DCF08	min DCF10	% EER	min DCF08	min DCF10
Interview in Train and Test, Same Microphone	1	1.62	79	402	1.62	84	379
Interview in Train and Test, Different Microphone	2	2.05	111	472	2.15	120	471
Interview in Train and Nve Phonecall Over Tel Channel in Test	3	2.02	122	443	2.77	128	463
Interview in Train and Nve Phonecall Over Mic Channel in Test	4	2.02	104	452	2.33	111	444
Nve Phonecall in Train and Test, Different Number	5	2.05	111	388	2.22	122	421
Nve Phonecall in Train and Hve Phonecall in Test, Tel	6	4.63	231	761	5.12	246	783
Nve Phonecall in Train and Hve Phonecall in Test, Mic	7	4.47	256	868	5.05	241	829
Nve Phonecall in Train and Lve Phonecall in Test, Tel	8	2.00	102	411	2.14	113	438
Nve Phonecall in Train and Lve Phonecall in Test, Mic	9	1.61	32	137	1.71	36	171
All	-	2.29	123	485	2.49	131	491

times heavily depend on the computer architecture, cache size, implementation language, and optimized numerical routines that are used. Thus, the relative speed–up of an approach with respect to the others is more meaningful and informative than the absolute i–vector extraction times. The eigen–decomposition approach has been chosen as the reference for the relative speed-up because it is the fastest among the ones considered in this work. Also, the memory values in [20] are doubled with respect to the ones reported for the experiments done in this work because in the latter we forced our Python implementation to use float rather than double precision structures. Using this configuration, the accuracy of the systems is practically unaffected, whereas relevant gain is obtained in terms of memory and speed–up.

The baseline results, corresponding to the standard i–vector extraction, were obtained 14 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 to store the terms $\mathbf{T}^{(c)*}\mathbf{T}^{(c)}$ required to speed–up the computation of (4).

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

Four implementation scenarios for the Variational Bayes approach [21], referred to as VB 10–10, VB 20–10, VB 10–100, and VB 20–100 in Table I were tested as part of the simulation experiments. Label VB b–t refers to the VB approach of [21] setting the sub–block dimension to b, and the stopping threshold to t. The stopping criterion is based on the difference between the L2-norm of the current estimated ivector

and the one computed in the previous iteration. The VB system with a tight convergence threshold is able to get the same results of the baseline systems, it is approximately 1.2 to 2 times slower than the standard approach, depending on the available number of concurrent threads, but it uses only slightly more memory than 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 due to a 10 times larger convergence threshold, leading to i–vector extraction which is at worst 1.3 times slower when using a single thread, but requires only 1/4 of memory required by the standard approach.

I–vector extraction with the FSE approach has been tested by training four systems, based on different dimensions of the dictionary: $K = 2000, 3500, 5000,$ and 10000, respectively. The i–vectors were obtained by the Conjugate Gradient procedure illustrated in Section VII, stopping the iterations when the residual $\mathbf{r}_n = \mathbf{c} - \hat{\mathbf{L}}\hat{\mathbf{w}}_n$ is less than two different thresholds, 100 or 10, respectively. Again two threshold values have been tested to show that the threshold value is not critical for the recognition accuracy. The results show that the FSE performance is always better than the eigen–decomposition approach, and depending on the dimension of the dictionary it can reach an accuracy comparable to the standard approach. FSE dramatically reduces the memory cost of i–vector extraction by 20 times compared to the standard approach, but also by 5 times compared to the other memory aware approaches. It is also extremely fast: faster than the standard method, and even faster than the eigen–decomposition approach for large UBM models and small dictionary size.

Comparing the results obtained with and without pre–conditioning the Conjugate Gradient, it can be observed that pre–conditioning contributes only a small speed–up for large models (5K and 10K), whereas it is detrimental for small models. Thus, it is not worth using pre–conditioning for small models because it does not speed up i–vector extraction, and requires $O(M)$ additional memory.

The small FSE-2K systems perform surprisingly well, considering that they use $1/5$ of the memory of the eigen–decomposition approach, but obtain results similar to the standard technique.

Dictionary dimensions smaller than $K = 2000$ are not included in Table I for two main reasons. First, our goal was to keep as much as possible the standard system performance, thus, we don't consider the results obtained with $K = 500$ and $K = 1000$, shown in Table II, comparable with the results given in Table I even for $K = 2000$. The second reason is that using such small values for K is not effective because memory occupation would be dominated by the set of matrices $\mathbf{O}^{(c)}$.

Since the relative performance of approximated approaches could be affected by the evaluation set that is used, we tested our FSE technique also on all conditions of the extended NIST SRE2010 male tests. Moreover, in order to maximize the difference with respect to the previous experiments, we used another

set of i-vectors obtained with different features, based on 45 Perceptual Linear Prediction, 19 Cepstrals (c0-c18), 19 delta ($\Delta_0 - \Delta_{18}$), and 7 delta–delta ($\Delta\Delta_0 - \Delta\Delta_6$), rather than on the 60 MFCC features used for the female tests.

The comparison of the performance of the standard and the Factorized Sub–space Estimation approach with a dictionary dimension of $K = 5000$ is shown in Table II. Excluding the tests with different vocal efforts, which are generally considered less significant, the DCFs of the FSE approach are at most 10% relative worse than the ones of the standard method, and decrease by 6.5% relative on the DCF08, and only 1.2% for the DCF10 on the overall test, as reported in the last row of Table II.

X. Conclusions

The aim of this work was to optimize the memory and computation time required for the i-vector extraction module of a speaker recognition system. A new approach has been presented that accurately approximates the components of the variability matrix by means of a linear combination of the atoms of a dictionary. The use of a common dictionary not only allows reducing the memory required with respect to the standard approach, but also with respect to the other memory–aware techniques, which cannot avoid storing the $\mathbf{T}$ matrix.

We analyzed the time and memory complexity of the state–of–the–art techniques and of our proposed method for i-vector extraction, and we also experimentally compared their performance. Our approach is not always as fast as the eigen–decomposition technique, but allows obtaining accurate i–vectors and results, and requires substantially less memory than the other techniques.

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. In fact, for short utterances, and for a given model dimension, the relative cost of i–vector extraction increases because the time devoted to i–vector extraction does not depend on the segment duration. A table giving the percentage of the overall recognition time devoted to i-vector extraction using the standard approach, as a function of the GMM dimensions, has been presented in Section VI-B of [21]. In these conditions, a fast but accurate technique, such as the FSE approach with a large enough dictionary, gives an important contribution to the reduction of the recognition times.

XI. Acknowledgments

We would like to thank Daniele Colibro and Claudio Vair from Nuance, for useful discussions and support, and the unknown reviewers for their valuable comments and suggestions.

APPENDIX

In this Appendix we prove that:

- The rank of $\mathbf{\Pi}^{(c)}$ in (12) is less than F if and only if two non–zero elements of $\mathbf{\Pi}^{(c)}$ appear in the same column.
- Step 2 and 3 of Algorithm 2 do not change the objective function (33).
- Step 3 of Algorithm 2 gives the optimal solution for merged row $\boldsymbol{\pi}_a^{(c)}$, thus the optimization of Step 4 can be performed simply on the cleared row $\boldsymbol{\pi}_b^{(c)}$.
- Algorithm 2 converges in a finite number of steps, providing a full–rank matrix $\mathbf{\Pi}^{(c)}$.

A. Rank of $\mathbf{\Pi}^{(c)}$

The procedure introduced in Section V-C for computing a new estimate of $\mathbf{\Pi}^{(c)}$, given the current estimate of $\mathbf{O}^{(c)}$ and $\mathbf{Q}$, does not guarantee that the new matrix $\mathbf{\Pi}^{(c)}$ is full rank, i.e., $\rho\left(\mathbf{\Pi}^{(c)}\right) = F$. In order to demonstrate that Algorithm 2 is able to increase the rank of $\mathbf{\Pi}^{(c)}$ we need to prove the following proposition:

Proposition 1:

Assume that $\rho(\mathbf{T}^{(c)}) = F$, $\rho(\mathbf{Q}) = M$ and that $\mathbf{\Pi}^{(c)}$ is computed as in Section V-C, i.e.,

$$\mathbf{\Pi}^{(c)} = \underset{\mathbf{\Pi}}{\operatorname{argmin}} \left\| \mathbf{T}^{(c)} - \mathbf{O}^{(c)} \mathbf{\Pi} \mathbf{Q} \right\|^2 .$$

Then

$$\rho\left(\mathbf{\Pi}^{(c)}\right) < F \Leftrightarrow \exists a, \exists b \quad | \quad k_a = k_b,\ |v_a| > 0, |v_b| > 0 \ ,$$

i.e., $\rho\left(\mathbf{\Pi}^{(c)}\right)$ is less than F if and only if two non–zero elements of $\mathbf{\Pi}$ appear in the same column.

Proof: The backward implication of the proposition is easily proved observing that whenever $k_a = k_b$ for rows $\boldsymbol{\pi}_a, \boldsymbol{\pi}_b$ of $\mathbf{\Pi}^{(c)}$, with values $|v_a| > 0$ and $|v_b| > 0$, respectively, the two rows are linearly dependent because $\boldsymbol{\pi}_a = \frac{v_a}{v_b} \boldsymbol{\pi}_b$.

As far as the forward implication of the proposition is concerned, let's assume that $k_a \neq k_b$ for all rows $\boldsymbol{\pi}_a$ and $\boldsymbol{\pi}_b$, and that $|v_f| > 0$ for all rows $\boldsymbol{\pi}_f$. Then, all rows of $\mathbf{\Pi}^{(c)}$ would be linearly independent, which would imply $\rho(\mathbf{\Pi}^{(c)}) = F$. Since this negates the hypothesis, then either a row f exists such that $\boldsymbol{\pi}_f = \mathbf{0}$, or a pair of rows $\boldsymbol{\pi}_a$ and $\boldsymbol{\pi}_b$ exist such that $k_a = k_b$. In order to prove our proposition, we show that $\boldsymbol{\pi}_f = \mathbf{0}$ for some f would contradict the hypothesis that $\boldsymbol{T}^{(c)}$ and $\mathbf{Q}$ are full rank matrices. Since,

according to the update rule for $\mathbf{\Pi}^{(c)}$ in (31), the value v_f, for each row f, is estimated as:

$$v_f = A^{(c)}_{f,k_f} \qquad k_f = \underset{k}{\operatorname{argmax}}\, {A^{(c)}_{f,k}}^2 \ ,$$

$v_f = 0$ implies that $A^{(c)}_{f,k} = 0$ for any choice of k. Since $\mathbf{A}^{(c)}$ would be a $F \times K$ matrix with a null row, we would have $\rho(\mathbf{A}^{(c)}) < F$. On the other hand, according to the definition of matrix $\mathbf{A}$ in (26), and since matrices $\mathbf{O}^{(c)}$, $\mathbf{T}^{(c)}$, and $\mathbf{Q}$ are full rank, matrix $\mathbf{A} = \mathbf{O}^{(\mathbf{c})*}\mathbf{T}^{(c)}\mathbf{Q}^*$ has rank F. Hence, $|v_f| > 0$ must be true for all f. Since $\rho\left(\mathbf{\Pi}^{(c)}\right) < F$ we can conclude that a pair of row indices a and b must exist such that $k_a = k_b$. ■

As a corollary of Proposition 1 we also obtain that $|v_f| > 0$ for all rows $\boldsymbol{\pi}_f$.

B. Step 2 and 3 do not change the objective function (33)

Let $\mathbf{\Pi}^{(c)}$ and $\mathbf{O}^{(c)}$ denote a (suboptimal) solution of our optimization problem, obtained by first optimizing with respect to $\mathbf{O}^{(c)}$ as described in Section V-B, and then with respect to $\mathbf{\Pi}^{(c)}$ as described in Section V-C. Let also $\boldsymbol{\pi}_a$ and $\boldsymbol{\pi}_b$ denote a pair of rows of $\mathbf{\Pi}^{(c)}$ such that $k_a = k_b = k$, and $g(\mathbf{O}^{(c)}, \mathbf{\Pi}^{(c)})$ denote the objective function related to the c–th component of $\mathbf{T}$, considering $\mathbf{Q}$ constant, defined in (33).

The first step to prove the convergence of the proposed algorithm consists in proving that

$$g(\bar{\mathbf{O}}^{(c)}, \bar{\mathbf{\Pi}}^{(c)}) = g(\mathbf{O}^{(c)}, \mathbf{\Pi}^{(c)}) \ ,$$

where $g(\bar{\mathbf{O}}^{(c)}, \bar{\mathbf{\Pi}}^{(c)})$ is the objective function value computed using the matrices $\bar{\mathbf{O}}^{(c)}$ and $\bar{\mathbf{\Pi}}^{(c)}$ defined in Algorithm 2 in Section VI. The two objective function values are equal because $\bar{\mathbf{O}}^{(c)}\bar{\mathbf{\Pi}}^{(c)} = \mathbf{O}^{(c)}\mathbf{\Pi}^{(c)}$. To prove latter statement, we rewrite $\bar{\mathbf{O}}^{(c)}\bar{\mathbf{\Pi}}^{(c)}$ as:

$$\bar{\mathbf{O}}^{(c)}\bar{\mathbf{\Pi}}^{(c)} = \sum_f \bar{\mathbf{o}}_f \bar{\boldsymbol{\pi}}_f \ , \tag{41}$$

where the sum extends over all columns $\bar{\mathbf{o}}_f$ of $\bar{\mathbf{O}}^{(c)}$ and all rows $\bar{\boldsymbol{\pi}}_f$ of $\bar{\mathbf{\Pi}}^{(c)}$. Further expanding (41), taking into account step 2 and step 3 of Algorithm 2 and the definitions in (34) we get:

$$\begin{aligned}
\bar{\mathbf{O}}^{(c)}\bar{\mathbf{\Pi}}^{(c)} &= \sum_f \bar{\mathbf{o}}_f \bar{\boldsymbol{\pi}}_f \\
&= \sum_{f \neq a,b} \bar{\mathbf{o}}_f \bar{\boldsymbol{\pi}}_f + \bar{\mathbf{o}}_a \bar{\boldsymbol{\pi}}_a + \bar{\mathbf{o}}_b \mathbf{0} \\
&= \sum_{f \neq a,b} \mathbf{o}_f \boldsymbol{\pi}_f + \frac{(v_a \mathbf{o}_a + v_b \mathbf{o}_b)}{\left(v_a^2 + v_b^2\right)^{\frac{1}{2}}} \left(v_a^2 + v_b^2\right)^{\frac{1}{2}} \mathbf{e}_k
\end{aligned}$$

$$= \sum_{f \neq a,b} \mathbf{o}_f \boldsymbol{\pi}_f + \mathbf{o}_a \mathbf{e}_k v_a + \mathbf{o}_b \mathbf{e}_k v_b$$
$$= \sum_{f} \mathbf{o}_f \boldsymbol{\pi}_f = \mathbf{O}^{(c)} \boldsymbol{\Pi}^{(c)} \ ,$$

where $\mathbf{e}_k$ is the k–th vector of the standard basis of $\mathbb{R}^K$, i.e., $\mathbf{e}_k = [0, \ldots, 0, 1, 0, \ldots, 0]$. The feasibility of the solution can be easily proved by observing that $\|\bar{\mathbf{o}}_a\| = 1$, $\|\bar{\mathbf{o}}_b\| = 1$ and $\bar{\mathbf{o}}_a^* \bar{\mathbf{o}}_b = 0$. Moreover, $\bar{\mathbf{o}}_a$ and $\bar{\mathbf{o}}_b$ are linear combinations of $\mathbf{o}_a$ and $\mathbf{o}_b$, and therefore are orthogonal to all other columns of $\bar{\mathbf{O}}^{(c)}$.

C. Step 3 gives the optimal solution for merged row $\boldsymbol{\pi}_a^{(c)}$

We prove that step 3 provides the optimal solution for the a–th row of matrix $\hat{\boldsymbol{\Pi}}^{(c)}$ so that matrices $\bar{\boldsymbol{\Pi}}^{(c)}$ and $\hat{\boldsymbol{\Pi}}^{(c)}$ differ only in their b–th rows. Let

$$\hat{\mathbf{A}}^{(c)} = \bar{\mathbf{O}}^{(c)*} \mathbf{T}^{(c)} \mathbf{Q}^* \ , \qquad \mathbf{A}^{(c)} = \mathbf{O}^{(c)*} \mathbf{T}^{(c)} \mathbf{Q}^* \ . \tag{42}$$

Since matrices $\bar{\mathbf{O}}^{(c)}$ and $\mathbf{O}^{(c)}$ differ only in their a–th and b–th columns, $\hat{\mathbf{A}}_f^{(\mathbf{c})} = \mathbf{A}_f^{(c)}$ for any row $f \neq a, b$.

The optimization in step 4 is performed for each row f of $\boldsymbol{\Pi}^{(c)}$ independently according to (31), we get, therefore, the same solution for any row $f \neq a, b$:

$$\hat{\boldsymbol{\pi}}_f = \boldsymbol{\pi}_f = \bar{\boldsymbol{\pi}}_f \quad \forall f \mid f \neq a, b \ . \tag{43}$$

To prove that also $\hat{\boldsymbol{\pi}}_a = \bar{\boldsymbol{\pi}}_a$ we rewrite the a–th row of $\hat{\mathbf{A}}^{(\mathbf{c})}$ as:

$$\hat{\mathbf{A}}_a^{(c)} = \frac{v_a \mathbf{A}_a^{(c)} + v_b \mathbf{A}_b^{(c)}}{\left(v_a^2 + v_b^2\right)^{\frac{1}{2}}} \ , \tag{44}$$

where the relation between $\mathbf{A}^{(c)}$ and $\hat{\mathbf{A}}_a^{(c)}$ derives from the relation between $\mathbf{O}^{(c)}$ and $\bar{\mathbf{O}}^{(c)}$ defined in step 2 of Algorithm 2. The optimal solution for the new row $\hat{\boldsymbol{\pi}}_a$ is given by:

$$k_a^{\text{opt}} = \underset{k}{\text{argmax}}\, \hat{A}_{a,k}^{(c)2} = \underset{k}{\text{argmax}} \frac{\left(v_a A_{a,k}^{(c)} + v_b A_{b,k}^{(c)}\right)^2}{\left(v_a^2 + v_b^2\right)}$$
$$v_{k_a^{\text{opt}}} = \hat{A}_{a,k^{\text{opt}}}^{(c)} = \frac{v_a A_{a,k^{\text{opt}}}^{(c)} + v_b A_{b,k^{\text{opt}}}^{(c)}}{\left(v_a^2 + v_b^2\right)^{\frac{1}{2}}} \ . \tag{45}$$

Since for every original row $\boldsymbol{\pi}_f^{(c)}$

$$v_f = A_{f,k_f}^{(c)}, \quad k_f = \underset{k}{\operatorname{argmax}} {A_{f,k}^{(c)}}^2 \ , \tag{46}$$

it follows that:

$$|v_a| = \max_k |A_{a,k}| \, , \qquad |v_b| = \max_k |A_{b,k}| \ , \tag{47}$$

thus,

$$\begin{aligned} \max_k \hat{A}_{a,k}^{(c)2} &= \max_k \frac{\left(v_a A_{a,k}^{(c)} + v_b A_{b,k}^{(c)}\right)^2}{\left(v_a^2 + v_b^2\right)} \\ &\leq \max_k \frac{\left(\left|v_a A_{a,k}^{(c)}\right| + \left|v_b A_{b,k}^{(c)}\right|\right)^2}{\left(v_a^2 + v_b^2\right)} \\ &\leq \frac{\left(|v_a| \max_{k_1} \left|A_{a,k_1}^{(c)}\right| + |v_b| \max_{k_2} \left|A_{b,k_2}^{(c)}\right|\right)^2}{\left(v_a^2 + v_b^2\right)} \\ &= \frac{\left(v_a^2 + v_b^2\right)^2}{\left(v_a^2 + v_b^2\right)} = \left(v_a^2 + v_b^2\right) \ . \end{aligned} \tag{48}$$

On the other hand, if we set $k_a^{\text{opt}} = k_a = k_b = k$, then from (45) we have:

$$\hat{A}_{a,k}^{(c)2} = \frac{\left(v_a^2 + v_b^2\right)^2}{\left(v_a^2 + v_b^2\right)} = \left(v_a^2 + v_b^2\right) \ . \tag{49}$$

Therefore, $k_a^{\text{opt}} = k_a = k_b = k$ is an optimal solution, and the corresponding value is given by:

$$v_{k_a^{\text{opt}}} = \left(v_a^2 + v_b^2\right)^{\frac{1}{2}} \ . \tag{50}$$

D. Convergence of Algorithm 2

Since, as previously shown, step 4 of the algorithm actually re–estimates only a single row $\hat{\boldsymbol{\pi}}_b$, which was $\mathbf{0}$, the corresponding contribution to the objective function is simply given by the value of (28) computed with the optimal value v_b obtained by (31), i.e. $\hat{A}_{b,k_b^{\text{opt}}}^{(c)2}$, thus the objective function decreases as:

$$g(\mathbf{O}^{(c)}, \boldsymbol{\Pi}^{(c)}) - g(\bar{\mathbf{O}}^{(c)}, \hat{\boldsymbol{\Pi}}^{(c)}) = \hat{A}_{b,k_b^{\text{opt}}}^{(c)2} \ . \tag{51}$$

The proposed algorithm terminates for two reasons: either $k_a \neq k_b$ for all rows, or two rows $\boldsymbol{\pi}_a$ and $\boldsymbol{\pi}_b$ exist such that $k_a = k_b$, but v_a or v_b or both are 0. However, Proposition 1 states that, if $\mathbf{T}^{(c)}$ and $\mathbf{Q}$

are full rank, step 4 estimates a row which has exactly one non–zero element. Thus, when Algorithm 2 terminates, every index k_f will refer to different rows $\boldsymbol{\pi}_f$ of $\mathbf{\Pi}^{(c)}$.

Finally, it remains to prove that the algorithm converges in a finite number of steps to a solution where $k_b \neq k_f$ for any $f \neq b$. The proof consist in showing that the objective function at every iteration of the algorithm never decreases less than a fixed value greater than zero.

Let's consider a generic orthogonal matrix $\mathbf{O}$, and the matrix product $\mathbf{O}^*\mathbf{T}^{(c)}\mathbf{Q}^*$. Let's also define δ as the smallest among the largest squared values of each row of this matrix. Formally:

$$\delta = \min_{\mathbf{O}} \min_{f} \max_{k} \left[\left(\mathbf{O}^*\mathbf{T}^{(c)}\mathbf{Q}^* \right)_{f,k} \right]^2 , \tag{52}$$

where the minimum and the maximum are taken across all rows and columns of the argument, respectively.

As a corollary of Proposition 1 it follows that, if $\mathbf{T}^{(c)}$ and $\mathbf{Q}$ are full rank matrices, $\delta > 0$. Since in step 4 we estimate a new solution minimizing the objective function (33) such that (51) is satisfied, and by definition

$$\hat{A}^{(c)}_{b,k^{\text{opt}}} = \max_{k} \left[\left(\bar{\mathbf{O}}^{(c)*}\mathbf{T}^{(c)}\mathbf{Q}^* \right)_{b,k} \right]^2 , \tag{53}$$

it follows that $\delta \leq \hat{A}^{(c)2}_{b,k^{\text{opt}}}$. Therefore:

$$g(\mathbf{O}^{(c)}, \mathbf{\Pi}^{(c)}) - g(\bar{\mathbf{O}}^{(c)}, \hat{\mathbf{\Pi}}^{(c)}) = \hat{A}^{(c)2}_{b,k^{\text{opt}}} \geq \delta \ . \tag{54}$$

Since the objective function is lower–bounded by zero, and at each iteration the objective function decreases by at least a finite positive amount δ, the algorithm terminates in a finite number of steps.

REFERENCES

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] N. Dehak, P. Torres-Carrasquillo, D. Reynolds, and R. Dehak, “Language recognition via i–vectors and dimensionality reduction,” in *Proceedings of Interspeech 2011*, pp. 857–860, 2011.

[9] G. Martinez, O. Plchot, L. Burget, O. Glembek, and P. Matějka, “Language recognition in i–vectors space,” in *Proceedings of Interspeech 2011*, pp. 861–864, 2011.

[10] C. Vaquero, A. Ortega, and E. Lleida, “Intra-session variability compensation and a hypothesis generation and selection strategy for speaker segmentation,” in *Proceedings of ICASSP 2011*, pp. 4532–4535, 2011.

[11] S. Shum, N. Dehak, E. Chuangsuwanich, D. Reynolds, and J. Glass, “Exploiting intra-conversation variability for speaker diarization,” in *Proceedings of INTERSPEECH 2011*, pp. 945–948, 2011.

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

[13] D. A. van Leeuwen, “Speaker linking in large data sets,” in *Proceedings of Odyssey 2010*, pp. 202–208, 2010.

[14] M. Kockmann, L. Burget, and J. H. Cernocky, “Application of speaker-and language identification state-of-the-art techniques for emotion recognition,” *Speech Communication*, vol. 53, no. 9–10, pp. 1172–1185, 2011.

[15] Xia and Y. Liu, “Using i-vector space model for emotion recognition,” in *Proceedings of Interspeech 2012*, 2012.

[16] M. H. Bahari, M. McLaren, H. V. Hamme, and D. A. van Leeuwen, “Age estimation from telephone speech using i-vectors,” in *Proceedings of Interspeech 2012*, 2012.

[17] 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.

[18] H. Aronowitz and O. Barkan, “Efficient approximated i–vector extraction,” in *Proceedings of ICASSP 2012*, pp. 4789–4792, 2012.

[19] P. Kenny, “A small footprint i-vector extractor,” in *Proceedings of Odyssey 2012*, pp. 1–6, 2012.

[20] S. Cumani, P. Laface, and V. Vasilakakis, “Memory and computation effective approaches for i–vector extraction,” in *Proceedings of Odyssey 2012*, pp. 7–13, 2012.

[21] S. Cumani and P. Laface, “Memory and computation trade-offs for efficient i-vector extraction,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 5, pp. 934–944, 2013.

[22] A. Larcher, P. Bousquet, K. A. Lee, D. Matrouf, H. Li, and J. Bonastre, “I-vectors in the context of phonetically-constrained short utterances for speaker verification,” in *Proceedings of ICASSP 2012*, pp. 4773–4776, 2012.

[23] S. Cumani, O. Plchot, and P. Laface, “Probabilistic Linear Discriminant Analysis of ivector posterior distributions,” in *Proceedings of ICASSP 2013*, pp. 7644–7648, 2011.

[24] P. Kenny, T. Stafylakis, P. Ouellet, M. Alam, and P. Dumouchel, “PLDA for speaker verification with utterances of arbitrary duration,” in *Proceedings of ICASSP 2013*, pp. 7649–7653, 2013.

[25] B. J. Borgstrom and A. McCree, “Discriminatively trained bayesian speaker comparison of i-vectors,” in *Proceedings of ICASSP 2012*, pp. 7659–7662, 2013.

[26] A. Aronowitz, “Text dependent speaker verification using a small development set,” in *Proceedings of Odyssey 2012*, pp. 312–316, 2012.

[27] A. Larcher, K. A. Lee, B. Ma, and H. Li, “Phonetically-constrained plda modeling for text-dependent speaker verification with multiple short utterances,” in *Proceedings of ICASSP 2013*, pp. 7673–7677, 2012.

[28] R. Rubinstein, A. Bruckstein, and M. Elad, “Dictionaries for sparse representation modeling,” *Proceedings of the IEEE*, vol. 598, no. 6, pp. 1045–1057, 2010.

[29] L. Mirsky, “A trace inequality of John von Neumann,” *Monatshefte fűr MathematikV*, vol. 79, no. 4, pp. 303–306, 2000.

[30] R. D. Grigorieff, “A note on von Neumann’s trace inequality,” *Math. Nachr.*, vol. 151, pp. 327–328, 1991.

[31] M. Hestenes and E. Stiefel, “Methods of conjugate gradients for solving linear systems,” *Journal of Research of the National Bureau of Standards*, vol. 59, no. 6, pp. 409–436, 1952.

[32] “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.

[33] 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.