SIMPLIFICATION AND OPTIMIZATION OF I-VECTOR EXTRACTION

Ondřej Glembek[1], *Lukáš Burget*[1], *Pavel Matějka*[1], *Martin Karafiát*[1], *Patrick Kenny*[2]

[1]Speech@FIT group, Brno University of Technology, Czech Republic
[2]Centre de Recherche Informatique de Montréal (CRIM), Montréal, Canada
{glembek,burget,matejkap,karafiat}@fit.vutbr.cz,
{patrick.kenny}@crim.ca

ABSTRACT

This paper introduces some simplifications to the i-vector speaker recognition systems. I-vector extraction as well as training of the i-vector extractor can be an expensive task both in terms of memory and speed. Under certain assumptions, the formulas for i-vector extraction—also used in i-vector extractor training—can be simplified and lead to a faster and memory more efficient code. The first assumption is that the GMM component alignment is constant across utterances and is given by the UBM GMM weights. The second assumption is that the i-vector extractor matrix can be linearly transformed so that its per-Gaussian components are orthogonal. We use PCA and HLDA to estimate this transform.

Index Terms— speaker recognition, i-vectors, Joint Factor Analysis, PCA, HLDA

1. INTRODUCTION

The i-vector systems have become the state-of-the-art technique in the speaker verification field [1]. They provide an elegant way of reducing the large-dimensional input data to a small-dimensional feature vector while retaining most of the relevant information. The technique was originally inspired by Joint Factor Analysis framework introduced in [2, 3].

The computational requirements for training the i-vector systems and estimating the i-vectors, however, are too high for certain types of applications. In this paper we propose simplifications to the original i-vector extraction and training schemes, which would dramatically decrease their complexity while retaining the recognition performance.

Our main motivation was running robust speaker verification systems on small scale devices such as mobile phones, as well as speeding up the process of speaker verification in real-time systems.

This paper is organized as follows: Section 2 introduces theoretical background of i-vector extraction and training of the i-vector extractor, Sections 3 and 4 introduce the proposed methods for i-vector extraction, Section 5 describes the experimental setup, Section 6 presents the recognition, speed, and memory performance, and Section 7 concludes the paper.

2. THEORETICAL BACKGROUND

Let us first state the motivation for the i-vectors. The main idea is that the speaker- and channel-dependent GMM supervector $\mathbf{s}$ can be modeled as:

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

where $\mathbf{m}$ is the UBM GMM mean supervector, $\mathbf{T}$ is a low-rank matrix representing M bases spanning subspace with important variability in the mean supervector space, and $\mathbf{w}$ is a standard normal distributed vector of size M.

For each observation $\mathcal{X}$, the aim is to estimate the parameters of the posterior probability of $\mathbf{w}$:

$$\mathrm{p}(\mathbf{w}|\mathcal{X}) = \mathcal{N}(\mathbf{w}; \mathbf{w}_{\mathcal{X}}, \mathbf{L}_{\mathcal{X}}^{-1}) \tag{2}$$

The i-vector is the MAP point estimate of the variable $\mathbf{w}$, i.e. the mean $\mathbf{w}_{\mathcal{X}}$ of the posterior distribution $\mathrm{p}(\mathbf{w}|\mathcal{X})$. It maps most of the relevant information from a variable-length observation $\mathcal{X}$ to a fixed- (small-) dimensional vector. $\mathbf{T}$ is referred to as the i-vector extractor.

2.1. Data

The input data for the observation $\mathcal{X}$ is given as a set of *zero-* and *first-order statistics* — $\mathbf{n}_{\mathcal{X}}$ and $\mathbf{f}_{\mathcal{X}}$. These are extracted from F dimensional features using a GMM UBM with C mixture components, defined by a mean supervector $\mathbf{m}$, component covariance matrices $\mathbf{\Sigma}^{(c)}$, and a vector of mixture weights $\boldsymbol{\omega}$. For each Gaussian component c, the statistics are given respectively as:

$$N_{\mathcal{X}}^{(c)} = \sum_t \gamma_t^{(c)} \tag{3}$$

$$\mathbf{f}_{\mathcal{X}}^{(c)} = \sum_t \gamma_t^{(c)} \mathbf{o}_t \tag{4}$$

where $\mathbf{o}_t$ is the feature vector in time t, and $\gamma_t^{(c)}$ is its occupation probability. The complete zero- and first-order statistics supervectors are $\mathbf{f}_{\mathcal{X}} = \left(\mathbf{f}_{\mathcal{X}}^{(1)\prime}, \ldots, \mathbf{f}_{\mathcal{X}}^{(C)\prime}\right)^\prime$, and $\mathbf{n}_{\mathcal{X}} = \left(N_{\mathcal{X}}^{(1)}, \ldots, N_{\mathcal{X}}^{(C)}\right)^\prime$.

For convenience, we *center* the first order statistics around the UBM means, which allows us to treat the UBM means effectively as a vector of zeros:

$$\begin{aligned} \mathbf{f}_{\mathcal{X}}^{(c)} &\leftarrow \mathbf{f}_{\mathcal{X}}^{(c)} - N_{\mathcal{X}}^{(c)} \mathbf{m}^{(c)} \\ \mathbf{m}^{(c)} &\leftarrow \mathbf{0} \end{aligned}$$

Similarily, we “normalize” the first-order statistics and the matrix $\mathbf{T}$ by the UBM covaricances, which again allows us to treat the UBM covariances as an identity matrix[1]:

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

[1] Part of the factor estimation is a computation of $\mathbf{T}'\mathbf{\Sigma}^{-1}\mathbf{f}$, where the decomposed $\mathbf{\Sigma}^{-1}$ can be projected to the neigboring terms, see [2] for detailed formulae.

978-1-4577-0539-7/11/$26.00 ©2011 IEEE

where $\mathbf{\Sigma}^{(c)-\frac{1}{2}}$ is a Cholesky decomposition of an inverse of $\mathbf{\Sigma}^{(c)}$, and $\mathbf{T}^{(c)}$ is a $F \times M$ sub-matrix of $\mathbf{T}$ corresponding to the c mixture component such that $\mathbf{T} = \left(\mathbf{T}^{(1)'}, \ldots, \mathbf{T}^{(C)'}\right)'$.

2.2. Parameter Estimation

As described in [2] and with the data transforms from previous section, for an observation $\mathcal{X}$, the corresponding i-vector is computed as a point estimate:

$$\mathbf{w}_{\mathcal{X}} = \mathbf{L}_{\mathcal{X}}^{-1}\mathbf{T}'\mathbf{f}_{\mathcal{X}} \tag{5}$$

where $\mathbf{L}$ is the precision matrix of the posterior distribution, computed as:

$$\mathbf{L}_{\mathcal{X}} = \mathbf{I} + \sum_{c=1}^{C} N_{\mathcal{X}}^{(c)}\mathbf{T}^{(c)'}\mathbf{T}^{(c)} \tag{6}$$

The computational complexity of the whole estimation for one observation is $O(CFM + CM^2 + M^3)$. The first term represents the $\mathbf{T}'\mathbf{f}_{\mathcal{X}}$ multiplication. The second term represents the sum in (6) and includes the multiplication of $\mathbf{L}_{\mathcal{X}}^{-1}$ with a vector. The third term represents the matrix inversion.

The memory complexity of the estimation is $O(CFM+CM^2)$. The first term represents the storage of all the input variables in (5), and the second term represents the pre-computed matrices in the sum of (6).

Note that the computation complexity grows quadratically with M in the sum of (6), and linearly with C. This becomes the bottleneck in the i-vector computation, resulting in high memory and CPU demands.

2.3. Model Training

Model hyper-parameters $\mathbf{T}$ are estimated using the same EM algorithm as in case of JFA [2]. Note that our algorithm makes use of an additional *minimum divergence* update step [3, 4], which yields a quicker convergence, but is not described here.

In the E step, the following accumulators are collected using all training observations i:

$$\mathbf{C} = \sum_i \mathbf{f}_i\mathbf{w}_i' \tag{7}$$

$$\mathbf{A}^{(c)} = \sum_i N_i^{(c)}\left(\mathbf{L}_i^{-1} + \mathbf{w}_i\mathbf{w}_i'\right) \tag{8}$$

where $\mathbf{w}_i$ and $\mathbf{L}_i$ are the estimates from (5) and (6) for observation i. The M step update is given as follows:

$$\mathbf{T}^{(c)} = \mathbf{C}\mathbf{A}^{(c)^{-1}} \tag{9}$$

3. SIMPLIFICATION 1: CONSTANT GMM COMPONENT ALIGNMENT

In this method, we apply the assumption that the GMM component alignment is constant across segments, i.e. the posterior occupation probabilities $\gamma^{(c)}$ in (3) are replaced by their prior probabilities represented by the UBM GMM weights. The new zero-order statistics are then:

$$\bar{N}_{\mathcal{X}}^{(c)} = \omega^{(c)}N_{\mathcal{X}} \tag{10}$$

where $\omega^{(c)}$ is the GMM UBM weight of component c, and $N_{\mathcal{X}} = \sum_{j=1}^{C} N_{\mathcal{X}}^{(j)}$. Substituting $N_{\mathcal{X}}^{(c)}$ in (6) by $\bar{N}_{\mathcal{X}}^{(c)}$ from (10), we get

$$\bar{\mathbf{L}}_{\mathcal{X}} = \mathbf{I} + N_{\mathcal{X}}\mathbf{W} \tag{11}$$

where

$$\mathbf{W} = \sum_{c=1}^{C} \omega^{(c)}\mathbf{T}^{(c)'}\mathbf{T}^{(c)} \tag{12}$$

Exploiting this simplification in the i-vector extractor training can be done at two stages: substituting $\mathbf{L}_i$ in (8) by (11), and substituting $N_i^{(c)}$ in (8) by (10). Based on our experiments, only the former turned out to be effective, therefore we will not report any results with the latter one.

Note that $\mathbf{W}$ in (12) is independent of data and can be pre-computed. Its resulting size is $M \times M$ yielding faster computation and less memory demands. The computational copmlexity of this algorithm reduces to $O(CFM + M^3)$ with the dominating inversion step. The memory complexity reduces to $O(CFM + M^2)$.

4. SIMPLIFICATION 2: I-VECTOR EXTRACTOR ORTHOGONALIZATION

Let us assume, that we can find a linear (orthogonal) transformation $\mathbf{G}$ which would orthogonalize all individual per-component sub-matrices $\mathbf{T}^{(c)}$. Orthogonalizing $\mathbf{T}$ would diagonalize $\mathbf{L}_{\mathcal{X}}$, which would need to be rotated back using $\mathbf{G}$. We can then express (6) as

$$\mathbf{L}_{\mathcal{X}} = \mathbf{G}^{(-1)'}\hat{\mathbf{L}}_{\mathcal{X}}\mathbf{G}^{-1} \tag{13}$$

where

$$\hat{\mathbf{L}}_{\mathcal{X}} = \mathbf{G}'\mathbf{G} + \sum_{c=1}^{C} N_{\mathcal{X}}^{(c)}\mathbf{G}'\mathbf{T}^{(c)'}\mathbf{T}^{(c)}\mathbf{G} \tag{14}$$

Assuming that $\hat{\mathbf{L}}_{\mathcal{X}}$ is diagonal, we can rewrite it as

$$\hat{\mathbf{L}}_{\mathcal{X}} = \mathrm{Diag}\left(\mathrm{diag}(\mathbf{G}'\mathbf{G}) + \mathbf{V}\mathbf{n}_{\mathcal{X}}\right) \tag{15}$$

where $\mathbf{V}$ is a $M \times C$ matrix whose cth column is $\mathrm{diag}(\mathbf{G}'\mathbf{T}^{(c)'}\mathbf{T}^{(c)}\mathbf{G})$. $\mathrm{Diag}(\cdot)$ maps a vector to a diagonal matrix, while $\mathrm{diag}(\cdot)$ maps a matrix diagonal to a vector. Combining (13) and (5), we get

$$\hat{\mathbf{w}}_{\mathcal{X}} = \mathbf{G}\hat{\mathbf{L}}_{\mathcal{X}}^{-1}\mathbf{G}'\mathbf{T}'\mathbf{f}_{\mathcal{X}} \tag{16}$$

The computational complexity of this approach is $O(CFM)$ as we can effectively simplify the matrix inversion to a vector element-wise inversion. The memory complexity is $O(CFM + M^2 + CM)$, where M^2 represents the extra diagonalization matrix $\mathbf{G}$, and CM represents $\mathbf{V}$ from (15).

The task is to estimate the orthogonalization matrix $\mathbf{G}$. Let us take a look at two approaches we investigated:

4.1. Eigen-decomposition

Let $\mathbf{W}$ be the weighted average per-component covariance matrix from (12). We assume $\mathbf{W}$ to be a full-rank matrix with M linearly independent eigenvectors. Then $\mathbf{W}$ can be factorized as

$$\mathbf{W} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} \tag{17}$$

where $\mathbf{Q}$ is a square $M \times M$ matrix whose ith column is the eigenvector $\mathbf{q}_i$ of $\mathbf{W}$ and $\mathbf{\Lambda}$ is a diagonal matrix whose diagonal elements are the corresponding eigenvalues. Matrix $\mathbf{Q}$ clearly orthogonalizes the space given by $\mathbf{W}$, therefore we can set $\mathbf{G} = \mathbf{Q}$.

4.2. Heteroscedastic Linear Discriminant Analysis

If the average covariance matrix $\mathbf{W}$ from (12) is close to diagonal, then the eigen-decomposition is not effective in diagonalizing the per-component covariances.

HLDA is a supervised method, which allows us to derive such projection that best de-correlates features associated with each particular class (maximum likelihood linear transformation for diagonal covariance modeling [5]). An efficient iterative algorithm [6] was used in our experiments to estimate matrix $\mathbf{G}$. In our task, the classes were defined as Gaussian mixture components. The within-class covariance matrices were given by $\mathbf{T}^{(c)\prime}\mathbf{T}^{(c)}$, and the occupation counts were provided as the mixture weights $\omega^{(c)}$.

Note that the well known Linear Discriminant Analysis (LDA) can be seen as special case of HLDA, where it is assumed that covariance matrices of all classes are the same.

5. EXPERIMENTAL SETUP

5.1. Feature Extraction

In our experiments, we used cepstral features, extracted using a 25 ms Hamming window. 19 Mel frequency cepstral coefficients together with log-energy were calculated every 10 ms. This 20-dimensional feature vector was subjected to short time mean and variance normalization using a 3s sliding window. Delta and double delta coefficients were then calculated using a 5-frame window giving 60-dimensional feature vectors.

Segmentation was based on the BUT Hungarian phoneme recognizer and relative average energy thresholding. Also, short segments were pruned out, after which the speech segments were merged together.

5.2. System Training

One gender-independent universal background model was represented as a diagonal covariance, 2048-component GMM. It was trained using LDC releases of Switchboard II, Phases 2 and 3; switchboard Cellular, Parts 1 and 2 and NIST 2004-2005 SRE.

One (gender-dependent) i-vector extractor was trained on the female part of the following telephone data: NIST SRE 2004, NIST SRE 2005, NIST SRE 2006, Switchboard II Phases 2 and 3, Switchboard Cellular Parts 1 and 2, Fisher English Parts 1 and 2 giving 8396 female speaker in 1463 hours of speech, and 6168 male speakers in 1098 hours of speech (both after voice activity detection).

Originally, 400 dimensional i-vector extractor was chosen as a reference. As mentioned later, training of the 800 dimensional system got feasible using one of the proposed methods. We trained such system to demonstrate the potentials of the proposed methods.

5.3. Scoring and Normalization

The same technique as in [1] was used. The extracted i-vectors were scaled down using an LDA matrix to 200 dimensions, and further normalized by a within-class covariance matrix. Both of these matrices were gender-dependent and were estimated on the same data as the i-vector extractor, except the Fisher data was excluded, resulting in 1684 female speakers in 715 hours of speech and 1270 male speakers in 537 hours of speech.

Cosine distance of the two input vectors was used as the raw score:

$$\operatorname{score}\left(\mathbf{w}_{\text{target}}, \mathbf{w}_{\text{test}}\right) = \frac{\langle \mathbf{w}_{\text{target}}, \mathbf{w}_{\text{test}} \rangle}{\|\mathbf{w}_{\text{target}}\| \|\mathbf{w}_{\text{test}}\|} \tag{18}$$

The cosine distance scores were normalized using gender-dependent s-norm [7] with a cohort of 400 speakers having 2 utterances per speaker.

5.4. Test Setup

The results of our experiments are reported on the female part of the Condition 5 (telephone-telephone) of the NIST 2010 speaker recognition evaluation (SRE) dataset [8]. The recognition accuracy is given as a set of equal error rate (EER), and the normalized DCF as defined both in the NIST 2010 SRE task (DCF_{new}) and the previous SRE evaluations (DCF_{old}).

The speed and memory performance of i-vector extraction were tested on a set of 50 randomly chosen utterances from the MIXER05 database. The input data (given as a set of fixed-size zero- and first-order statistics) and all of the input parameters were included in the general memory requirements. The following algorithm-specific terms were pre-computed (thus not included in the reported times), and comprised in the algorithm-specific memory requirements:

- $\mathbf{T}^{(c)\prime}\mathbf{T}^{(c)}$ in (6)
- $\mathbf{W}$ in (12)
- $\mathbf{G}$ and $\mathbf{T}^{(c)}\mathbf{G}$ in (13) and (16), and $\mathbf{V}$ in (15)

The algorithms were tested in MATLAB (R2009b) 64-bit, running in a single thread and the default double-precision mode. The machine was an Intel(R) Xeon(R) CPU X5670 2.93GHz, with 36GB RAM.

6. RESULTS

In the following section, we will reference the systems according to the i-vector dimensionality and to the extraction method used. *Baseline* stands for the original method as in Sec. 2.2, and *simple 1* and *simple 2* reference to the proposed simplifications.

Table 1 summarizes the systems with respect to verification accuracy. Fig. 1 visualizes the different systems on a constellation plot. The “800 baseline” system is clearly the winner, however “800 simple 2 - HLDA” is a tight competitor to the “400 baseline”.

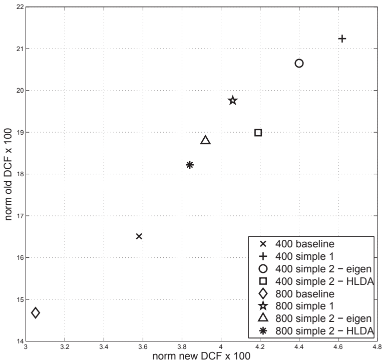

Fig. 1. Constellation plot of the individual systems

6.1. Speed and Memory

As described earlier in Sec. 5.4, the computation time does not include reading of the necessary data and pre-computation of some terms. The results are reported in Tab. 2. The dominating complexity of matrix inversion makes “simple 2” faster than “simple 1”, as described in Sec. 3 and 4.

Table 1. *Comparison of the proposed i-vector extraction methods in terms of normalized DCFs and EER*

	$\mathrm{DCF}_{\mathrm{new}}$	$\mathrm{DCF}_{\mathrm{old}}$	EER
400 baseline	0.5395	0.1651	3.58
400 simple 1	0.6664	0.2124	4.62
400 simple 2 - eigen	0.6627	0.2065	4.40
400 simple 2 - HLDA	0.6236	0.1899	4.19
800 baseline	0.4956	0.1468	3.05
800 simple 1	0.6057	0.1976	4.06
800 simple 2 - eigen	0.5414	0.1879	3.92
800 simple 2 - HLDA	0.5694	0.1822	3.84

Table 2. *Comparison of the proposed i-vector extraction methods in processing speed.*

	absolute [sec]	relative to 400 baseline
400 baseline	13.70	100.00%
400 simple 1	1.01	7.37%
400 simple 2	0.54	3.94%
800 baseline	65.75	480.00%
800 simple 1	3.64	26.57%
800 simple 2	1.11	8.10%

Tab. 3 shows memory allocation for different systems. We see that for most of the current hardware configurations, the baseline systems could be a problem.

Table 3. *Comparison of the proposed i-vector extraction methods in memory allocation (in MB). The “constant” term depends on the i-vector dimensionality.*

	constant	algorithm specific	total
400 baseline	422.96	2,500.00	2,923.00
400 simple 1	”	1.22	424.18
400 simple 2	”	7.47	430.43
800 baseline	802.84	10,000.00	10,802.84
800 simple 1	”	4.88	807.83
800 simple 2	”	17.38	820.23

Note that prior to the scoring, WCCN and LDA dimensionality reduction are applied to the i-vectors (see Sec. 5.3). Projecting this linear transformation directly into the leftmost $\mathbf{G}$ of (16) could further decrease the complexity of the “simple 2” algorithm.

6.2. Simplification 1 in Training

While none of the simplifications had positive contribution to the test accuracy, the training phase simplification results in negligible accuracy changes while exploiting some of the speed and memory advantages as described in the previous section. Table 4 shows the difference.

Time and memory complexity of collecting the accumulators $\mathbf{A}$ from (8) is almost identical to the computation of $\mathbf{L}_{\mathcal{X}}$ in (6). The proposed method still keeps the same accumulator collection, however, avoiding the expensive computation of (6) decreases the E step time and memory complexity by a factor of 2.

Table 4. *Comparison of the proposed i-vector extractor training methods in terms of normalized DCFs and EER*

	$\mathrm{DCF}_{\mathrm{new}}$	$\mathrm{DCF}_{\mathrm{old}}$	EER
400 baseline	0.5460	0.1722	3.40
400 simple 1	0.5376	0.1729	3.42

7. CONCLUSIONS

We managed to reduce the memory requirements and processing time for the i-vector extractor training so that higher dimensions can be now used while retaining the recognition accuracy. As for i-vector extraction, we managed to reduce the complexity of the algorithm with sacrificing little recognition accuracy, which makes this technique usable in small-scale devices.

As a practical result, Simplification 1 was used in the MOBIO project, when porting a speaker verification system on a mobile phone platform.

Not only we managed to scale down the complexity of the system in terms of real-world applications, but also we have prepared a set of simplified formulas which could potentially find use in a future research, such as discriminative training.

8. ACKNOWLEDGMENTS

The work was partly supported by European project MOBIO (FP7-214324), Grant Agency of Czech Republic project No. 102/08/0707, Czech Ministry of Education project No. MSM0021630528 and by BUT FIT grant No. FIT-10-S-2. Great part of the work was done at the BOSARIS workshop held at BUT in July 2010.

9. REFERENCES

[1] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, “Front-end factor analysis for speaker verification,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. PP, no. 99, 2010.

[2] P. Kenny, “Joint factor analysis of speaker and session variability : Theory and algorithms - technical report CRIM-06/08-13. Montreal, CRIM, 2005,” 2005.

[3] P. Kenny, G. Boulianne, P. Oullet, and P. Dumouchel, “Joint factor analysis versus eigenchannes in speaker recognition,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2072–2084, 2007.

[4] Niko Brümmer, “The EM algorithm and minimum divergence,” Agnitio Labs Technical Report. Online: http://niko.brummer.googlepages.com/EMandMINDIV.pdf, Oct. 2009.

[5] N. Kumar, *Investigation of Silicon-Auditory Models and Generalization of Linear Discriminant Analysis for Improved Speech Recognition*, Ph.D. thesis, John Hopkins University, Baltimore, 1997.

[6] M.J.F. Gales, “Semi-tied covariance matrices for Hidden Markov Models,” *IEEE Trans. Speech and Audio Processing*, vol. 7, pp. 272–281, 1999.

[7] N. Brümmer and A. Strasheim, “AGNITIO’s speaker recognition system for EVALITA 2009,” 2009.

[8] “National institute of standard and technology,” http://www.nist.gov/speech/tests/spk/index.htm.