VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OPTIMALIZACE MODELOVÁNÍ GAUSSOVSKÝCH SMĚSÍ V PODPROSTORECH A JEJICH SKÓROVÁNÍ V ROZPOZNÁVÁNÍ MLUVČÍHO

OPTIMIZATION OF GAUSSIAN MIXTURE SUBSPACE MODELS
AND RELATED SCORING ALGORITHMS IN SPEAKER VERIFICATION

DISERTAČNÍ PRÁCE

PHD THESIS

AUTOR PRÁCE — Ing. ONDŘEJ GLEMBEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. LUKÁŠ BURGET, Ph.D.

SUPERVISOR

BRNO 2012

Abstract

This thesis deals with Gaussian Mixture Subspace Modeling in automatic speaker recognition. The thesis consists of three parts. In the first part, Joint Factor Analysis (JFA) scoring methods are studied. The methods differ mainly in how they deal with the channel of the tested utterance. The general JFA likelihood function is investigated and the methods are compared both in terms of accuracy and speed. It was found that linear approximation of the log-likelihood function gives comparable results to the full log-likelihood evaluation while simplyfing the formula and dramatically reducing the computation speed.

In the second part, i-vector extraction is studied and two simplification methods are proposed. The motivation for this part was to allow for using the state-of-the-art technique on small scale devices and to setup a simple discriminative-training system. It is shown that, for long utterances, while sacrificing the accuracy, we can get very fast and compact i-vector systems. On a short-utterance(5-second) task, the results of the simplified systems are comparable to the full i-vector extraction.

The third part deals with discriminative training in automatic speaker recognition. Previous work in the field is summarized and—based on the knowledge from the earlier chapters of this work—discriminative training of the i-vector extractor parameters is proposed. It is shown that discriminative re-training of the i-vector extractor can improve the system if the initial estimation is computed using the generative approach.

Keywords

Speaker Recognition, Gaussian Mixture Model, Subspace Modeling, i-vector, Joint Factor Analysis, Discriminative Training

Bibliographic citation

Ondřej Glembek: Optimization of Gaussian Mixture Subspace Models and Related Scoring Algorithms in Speaker Verification, **Doctoral thesis**, **Brno**, **Brno University of Technology, Faculty of Information Technology**, 2012

Abstrakt

Tato práce pojednává o modelování v podprostoru parametrů směsí gaussovských rozložení pro rozpoznávání mluvčího. Práce se skládá ze tří částí. První část je věnována skórovacím metodám při použití sdružené faktorové analýzy k modelování mluvčího. Studované metody se liší převážně v tom, jak se vypořádávají s variabilitou kanálu testovacích nahrávek. Metody jsou prezentovány v souvislosti s obecnou formou funkce pravděpodobnosti pro sdruženou faktorovou analýzu a porovnány jak z hlediska přesnosti, tak i z hlediska rychlosti. Je zde prokázáno, že použití lineární aproximace pravděpodobnostní funkce dává výsledky srovnatelné se standardním vyhodnocením pravděpodobnosti při dramatickém zjednodušení matematického zápisu a tím i zvýšení rychlosti vyhodnocování.

Druhá část pojednává o extrakci tzv. i-vektorů, tedy nízkodimenzionálních reprezentací nahrávek. Práce prezentuje dva přístupy ke zjednodušení extrakce. Motivací pro tuto část bylo jednak urychlení extrakce i-vektorů, jednak nasazení této úspěšné techniky na jednoduchá zařízení typu mobilní telefon, a také matematické zjednodušení umožněňující využití numerických optimalizačních metod pro diskriminativní trénování. Výsledky ukazují, že na dlouhých nahrávkách je zrychlení vykoupeno poklesem úspěšnosti rozpoznávání, avšak na krátkých nahrávkách, kde je úspěšnost rozpoznávání nízká, se rozdíly úspěšnosti stírají.

Třetí část se zabývá diskriminativním trénováním v oblasti rozpoznávání mluvčího. Jsou zde shrnuty poznatky z předchozích prací zabývajících se touto problematikou. Kapitola navazuje na poznatky z předchozích dvou částí a pojednává o diskriminativním trénování parametrů extraktoru i-vektorů. Výsledky ukazují, že při klasickém trénování extraktoru a následném diskriminatviním přetrénování tyto metody zvyšují úspěšnost.

Klíčová slova

rozpoznávání mluvčího, směs gaussovských rozložení, modelování v podprostoru parametrů, i-vector, sdružená faktorová analýza, diskriminativní trénování

Bibliografická citace

Ondřej Glembek: Optimalizace modelování gaussovských směsí v podprostorech a jejich skórování v rozpoznávání mluvčího, **Disertační práce**, **Brno**, **Vysoké učení technické v Brne, Fakulta informačních technologií**, 2012

Chapter 1

Introduction

Automatic speaker recognition (SRE) is the process of classifying audio recording based on the information which is relevant to the speaker in that recording. It is assumed that the process is independent of the *channel*, i.e. language, communication channel, content, etc. The problem can be understood from two points of view: *speaker identification*, and *speaker verification.*

Speaker identification is a multi-class classification problem, where the task is to assign an utterance to a closed set of known speaker labels. An example of such application could be a search engine in an audio database of university lecture recordings. If a new recording by a staff member is to be added to the database, the speaker can be automatically identified assigned to the new recording. Note that this approach fails if the speaker is a guest and his *voiceprint* is not in the database of known speakers.

Speaker verification—on the other hand—is a two-class problem, where the task is to decide whether two utterances come from the same speaker or not. This task is sometimes reinterpreted as to decide whether an utterance belongs to a certain speaker model or not. Since the speaker model is assumed to have been computed from some reference utterance, the two interpretations of the problem are equivalent. An example of such application could be e.g. telephone-banking authentication, where—apart from answering questions about e.g. mother's maiden name, date of birth, social security number, etc.—the voiceprint match gives yet another level of security. Speaker verification can be easily converted to speaker identification by restricting the set of compared utterances.

Looking at the SRE problem content-wise, we can understand it as either *text-dependent* or *text-independent.* While text-dependent SRE looks at the content of the speech, such as passphrase, the text-independent approach only exploits the information in the waveform, basically ignoring what is being said. Looking at the possible scenarios, text-dependent SRE system could be employed in a telebanking system where the user authenticates using a passphrase that only he or she is supposed to know, while text-independent system is more suitable for intelligence purposes, such as spotting a suspicious person on a telephone network. The point of interest of this work is *text-independent speaker verification.*

1.0.1 Voice Activity Detection

In most speech-processing applications including SRE, Voice Activity Detection (VAD) is run to choose the parts of the analyzed utterance, which do contain useful speech. There are various approaches to this step including mere energy thresholding, Gaussian Mixture Model (GMM) approaches to advanced and robust Neural Networks (NN) and Hidden Markov Models (see e.g. system descriptions of [NIST, nda]).

In this work, VAD is based on hybrid Artificial Neural Networks (ANN) / Hidden Markov Model (HMM). It is used as phoneme recognizer trained on the SPEECHDAT Hungarian database [Matějka et al., 2006]. The output of such recognizer is a string of recognized phonemes in the analyzed utterances. The phonemes are then clustered into two classes—silence (all models of silence) and speech (all valid speech phonemes).

In case of telephone conversations, the cross-talks are detected by comparing the speech energies in the overlapping string transcriptions (e.g. [Matějka et al., 2006]. The segment with the higher energy is considered as the original channel.

1.0.2 Feature Extraction

In the orders of milliseconds, the acoustic signal can be considered stationary. This assumption allows to split the signal into short (typically 10ms) units referred to as *frames*. This operation can be viewed as *windowing* of the signal by a square window function. The cuts at the borders of the frames introduce high-frequency distortion, therefore the rectangular window function is usually substituted with a bell-shaped *Hamming-window* function [Young et al., 2006], which attenuates the border area of the window and therefore suppresses the unwanted distortion. The drawback is that the useful information in the border area is also suppressed, therefore the window function is usually set longer than the windows shift and the windows overlap. To summarize the procedure, the typical scenario is that frames are extracted every 10ms and their usual length is 20–25ms.

Speech information is extracted from the frames in the form of *feature vectors*. A feature vector is a low-dimensional representation of a speech frame. In this work we have used the Mel-Filterbank Cepstral Coefficients with various post-processing steps.

Feature Derivatives

To capture the time progression, the consecutive feature vectors are usually extended with their 1^{st}, 2^{nd}, and/or 3^{rd} order derivative approximations (higher orders are rarely used), commonly referred to as deltas, double-deltas, and triple-deltas [Furui, 1986].

Feature Normalization

It has been observed that it is common for the dynamics of the features to vary from one utterance to another in a linear way, i.e. they can get biased and scaled. To deal with this sort of *inter-session variation* on the feature level, feature mean removal and variance scaling has been proposed (e.g. [Young et al., 2006]); for a k-th frame in utterance d, the normalized i-th coefficient $\hat{c}_{d,i}(k)$ is computed as

$$\hat{c}_{d,i}(k) = \frac{c_{d,i}(k) - \mu_{d,i}}{\sigma_{d,i}}, \tag{1.1}$$

where the normalization parameters mean $\mu_{d,i}$ and standard deviation $\sigma_{d,i}$ are usually calculated from the whole utterance d.

Short-Time Normalization

It is also common to compute the normalization parameters on short segments and apply them locally. For each frame, the scale and bias is computed from a short segment centered around the frame. This operation compensates for the within-session variability and has proved to be effective for some SRE systems. The typical length of such short-window is 3s.

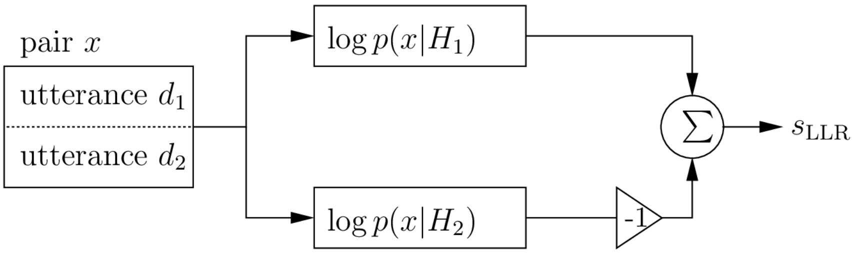


Figure 1.1: General symmetrical speaker verification procedure: an input trial x is given as a pair of utterances $\langle d_1, d_2\rangle$ and the computation of the likelihood is conditioned by the hypotheses that the utterances come either from a single speaker ($\mathcal{H}_1$) or two different speakers ($\mathcal{H}_2$).

Short-Time Gaussianization

Not only can the features be normalized locally as mentioned in the previous section, they can also be warped to have standard normal distribution. This step is known as Feature Warping or Short-Time Gaussianization (STG) and has been proposed in [Pelecanos and Sridharan, 2006]. The algorithm operates on a short window of features (typically 3s). It sorts the features and substitutes them with corresponding values of an inverse cumulative density function (CDF) of a standard normal distribution.

1.1 Automatic Speaker Verification Procedure

As was said in the introduction, the task of speaker verification is to detect whether a pair of utterances comes from the same speaker (referred to as hypothesis $\mathcal{H}_1$) or from different speakers (hypothesis $\mathcal{H}_2$). It is assumed in this work, that the utterances themselves do not contain speech from multiple speakers.

The detection is based on evaluating statistical models using the data provided. It is realized using a 2-class classifier whose outcome is a log-likelihood ratio (LLR) of the two hypotheses. Following the definition of speaker verification from the previous paragraph, the diagram of speaker verification procedure can be seen in Figure 1.1. Note that the input to the likelihood function is a pair of two utterances d_1 and d_2—referred to as a *trial* $x = \langle d_1, d_2\rangle$—and the system is generally symmetrical, i.e. the order of d_1 and d_2 does not matter. Mathematically, the score is given as

$$s_{\mathrm{sym}} = \log \frac{p(x|\mathcal{H}_1)}{p(x|\mathcal{H}_2)} \tag{1.2}$$

However, the problem has traditionally been seen as a two-phase asymmetrical task. In the first phase of *enrollment*, a speaker model $\mathcal{M}_1$ has to be trained from the utterance d_1 (referred to as the enrollment data). Then, in the *scoring* phase, the score is computed as a ratio of how likely the utterance d_2 (referred to as *test* data) is generated by $\mathcal{M}_1$ and how likely it is generated by "any" speaker model $\mathcal{M}_{\mathrm{UBM}}$, where UBM stands for Universal Background Model. The UBM approximates the distribution of all speakers. To do so, it is trained on some training set comprising many speakers. A diagram of such evaluation is shown in Figure 1.2. Mathematically, the asymmetrical problem is stated as

$$s_{\mathrm{assym}} = \log \frac{p(d_2|\mathcal{M}_2)}{p(d_2|\mathcal{M}_{\mathrm{UBM}})} \tag{1.3}$$

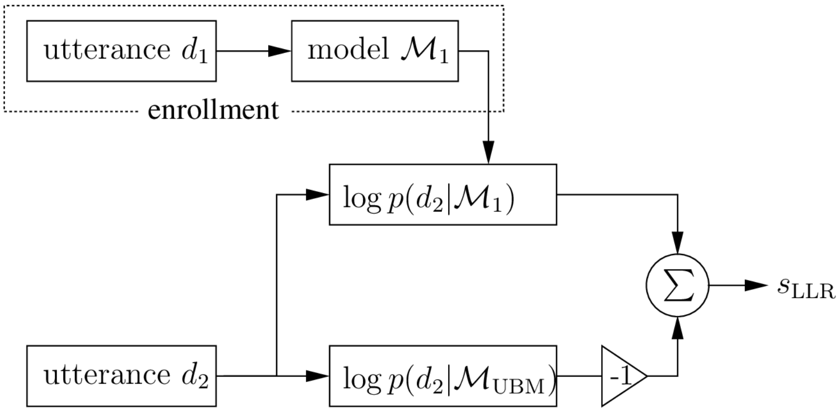


Figure 1.2: Asymmetrical speaker verification procedure: an input trial x is given as a model $\mathcal{M}_1$ and test utterance d_2 and the computation of the likelihood d_2 is conditioned by the model, i.e. either the probed speaker model $\mathcal{M}_1$ or the "the model of all speakers"—the UBM.

In this work, both described definitions are used, depending on the overall modeling method used.

1.1.1 Model Training

As mentioned, the detection is based on evaluating statistical models. Essentially, there are two approaches to modeling: *generative* and *discriminative*. The generative approach aims at training the models so that they are most likely to have generated the input data. This approach has traditionally been used due to its simplicity and flexibility. Note, that the description of this problem does not mention anything about classes or classification. A common method for estimating model parameters is e.g. maximum likelihood.

On the other hand, discriminative training aims at training the parameters so that, when applied, they address the problem of class separation in a direct way. In speaker recognition, discriminative training has originally been based on Support Vector Machines (SVM) [Vapnik, 1995, Campbell, 2002, Campbell et al., 2006]. SVMs were used in place of $\mathcal{M}_1$ of Figure 1.2 and they were trained for each speaker to best match the speaker against a cohort of impostors. The problem was therefore defined as one against many. In this work—as studied in Chapter 4.2—discriminative training will address SRE as a symmetrical problem (as shown in Figure 1.1). The input will be a pair of tested utterances and the classes will be given by the same-speaker and different-speaker hypotheses, i.e. $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively.

1.1.2 Score Normalization

It has been shown in [Auckenthaler et al., 2000], that score normalization compensates for data mismatch. The assumption is that the impostor scores are normally distributed and the principle of score normalization is to apply scaling and shift to force the impostor scores to be of standard normal distribution. The scale and shift are estimated using separate normalization sets which are assumed to contain recordings from impostor speakers only. The scale and shift is applied as

$$s_{\mathrm{norm}} = \frac{s-\mu}{\sigma}. \tag{1.4}$$

Zero Normalization—Z-norm

Assuming the asymmetrical approach, this method estimates the normalization constants by having a set of impostor recordings Z scored against the enrolled speaker model $\mathcal{M}$. Mathematically, we assure that

$$p\left(\frac{s_{\text{imp}} - \mu_{\mathcal{M}}}{\sigma_{\mathcal{M}}}\middle|\mathcal{M}\right) = \mathcal{N}\left(\frac{s_{\text{imp}} - \mu_{\mathcal{M}}}{\sigma_{\mathcal{M}}}; 0, 1\right) \tag{1.5}$$

Figure 1.3 depicts this procedure as "STEP 1". This normalization compensates for the acoustic mismatch between the set of "standard" test utterances and the data that were used to train the speaker model. The advantage of Z-norm is that the estimation of the constants can be performed off-line when enrolling the model.

Test Normalization—T-norm

This method is similar to Z-norm in that a mean is subtracted and score is scaled by a standard deviation. When scoring an utterance d, a set of impostor models M is used to compute the parameters, which are then applied using (1.4). Mathematically, we assure that

$$p\left(\frac{s_{\text{imp}} - \mu_d}{\sigma_d}\middle|d\right) = \mathcal{N}\left(\frac{s_{\text{imp}} - \mu_d}{\sigma_d}; 0, 1\right) \tag{1.6}$$

The method is marked as "STEP 3" in Figure 1.3. It compensates for the acoustic mismatch between the tested utterance and a set of "standard" speaker models.

ZT-norm

ZT-norm is a combination of both normalization techniques. For simplicity, let us assume that we have a matrix of all scores, where each row corresponds to an enrolled model and each column to a tested utterance. First, the Z-norm is applied to the matrix of test scores and to the matrix of scores of T-norm models vs. test utterances, denoted as STEP 1 and STEP 2 in Figure 1.3. Next, T-norm parameters are computed on the T-norm–test matrix, and applied to the matrix of (Z-normalized) test scores, denoted as STEP 3 in Figure 1.3.

S-norm

S-norm has been introduced for the symmetrical systems as the Z- or T- norm concept is asymmetrical by nature. The S-norm that was used in this work is basically computed as the average of Z- and T-norm scores, where the cohorts are the same.

1.2 Motivation and Contribution

The first part of my work was done during the John Hopkins University 2008 summer workshop [Burget et al., 2008], which consisted mainly in comparison of different scoring methods for Joint Factor Analysis. My interest was to compare the methods and analyze them in deep [Glembek et al., 2009]. During the workshop, Najim Dehak invented the i-vectors [Dehak et al., 2010], which outperformed JFA and had quickly become the essence of the modern SRE systems. The second part of my work was inspired by Najim's work and the on-going Mobio project [Marcel et al., 2010], one of whose aim was to implement speaker verification on a cell-phone. I was interested in simplifying the i-vector extraction so that it could be used in Mobio. The underlying work was presented in [Glembek et al., 2011b]. At that

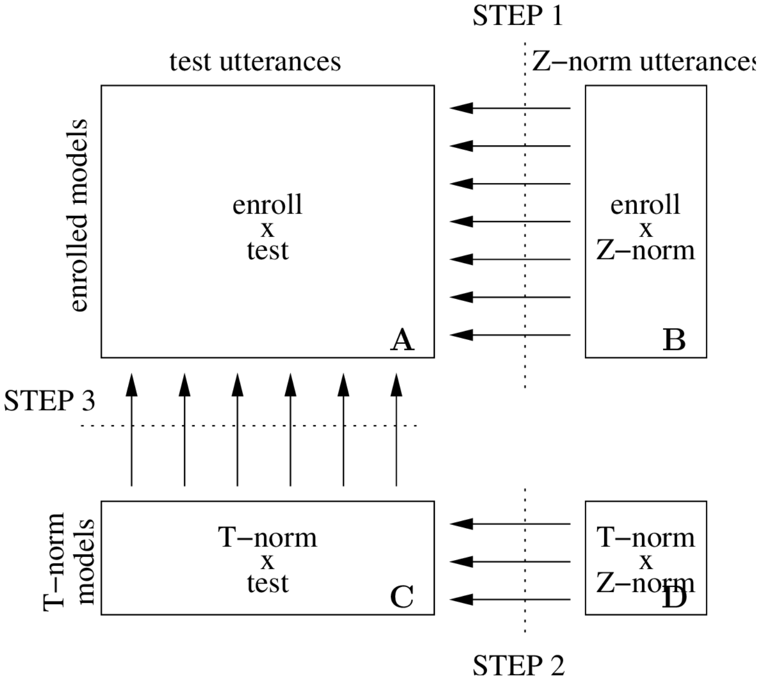


Figure 1.3: Application of ZT-norm. The boxes denote matrices of complete scores, i.e. all models against all scored utterances.

very same JHU workshop, Lukas Burget tried to train the JFA discriminatively. Later on, he experimented with discriminatively optimizing the PLDA [Burget et al., 2011]. The third part of my work was inspired by Lukas' work and I have tried to apply the discriminative training framework to the i-vector system [Glembek et al., 2011a].

1.2.1 Claims

The goal of this work was to analyze the contemporary state-of-the-art speaker recognition systems and to improve the methods not only in terms of accuracy, but also in terms of speed and real-world application.

- **analysis of JFA scoring methods**: I systematically investigated different scoring methods for JFA that different sites have been using and analyzed them in terms of anatomy, speed, and accuracy.
- **i-vector extraction optimization**: The computational requirements for training the i-vector systems and estimating the i-vectors, are too high for certain types of applications. In this work I introduce simplifications to the original i-vector extraction and training schemes, which dramatically decrease their complexity while retaining the recognition performance.
- **i-vector extractor training simplification**: Using the new proposed method, larger i-vector systems can be trained as memory demands have halved.
- **discriminative training of i-vector extractor**: I have implemented and tested the i-vector extractor training using discriminative criterion. The approach was tested on a scaled-down system and shown an improvement for the simplified i-vector extraction.

Chapter 2

Joint Factor Analysis

Joint Factor Analysis (JFA) is a GMM subspace modeling technique, which has been proposed to model the speaker *and* session variabilities. It has undergone a series of modifications and has attracted many researcher's attention resulting in numerous interesting publications. However, when comparing their results, people used different functions to obtain the score ([Kenny et al., 2007], [Vair et al., 2007], [Brümmer et al., 2007]). This chapter gives a brief introduction to JFA, mostly from a practical point of view, i.e. it concentrates on explaining how the model parameters are trained and how the score is estimated with respect to the paradigms of JFA.

2.1 Theoretical background

Joint factor analysis is a model used to treat the problem of speaker and session variability in GMMs. In this model, each speaker is represented by the means, covariance, and weights of a mixture of C multivariate Gaussian densities defined in some continuous feature space of dimension F. The GMM for a target speaker is obtained by adapting the Universal Background Model (UBM) mean parameters. In Joint Factor Analysis [Kenny et al., 2007], the basic assumption is that a speaker- and channel- dependent supervector of means $\mathbf{M}$ can be decomposed into a sum of two supervectors: a speaker supervector $\mathbf{s}$ and a channel supervector $\mathbf{c}$

$$\mathbf{M} = \mathbf{s} + \mathbf{c}, \tag{2.1}$$

where $\mathbf{s}$ and $\mathbf{c}$ are normally distributed. In [Kenny et al., 2008], Kenny et al. described how the speaker dependent supervector and channel dependent supervector can be represented in low dimensional spaces. The first term in the right hand side of (2.1) is modeled by assuming that if $\mathbf{s}$ is the speaker supervector for a randomly chosen speaker then

$$\mathbf{s} = \mathbf{m} + \mathbf{V}\mathbf{y} + \mathbf{D}\mathbf{z}, \tag{2.2}$$

where $\mathbf{m}$ is the speaker and channel independent supervector (UBM), $\mathbf{D}$ is a diagonal matrix, $\mathbf{V}$ is a rectangular matrix of low rank and $\mathbf{y}$ and $\mathbf{z}$ are independent random vectors having standard normal distributions. In other words, $\mathbf{s}$ is assumed to be normally distributed with mean $\mathbf{m}$ and covariance matrix $\mathbf{V}\mathbf{V}^* + \mathbf{D}\mathbf{D}^*$. The components of $\mathbf{y}$ and $\mathbf{z}$ are respectively the speaker and common *factors*.

The channel-dependent supervector $\mathbf{c}$, which represents the channel effect in an utterance, is assumed to be distributed according to

$$\mathbf{c} = \mathbf{U}\mathbf{x}, \tag{2.3}$$

where $\mathbf{U}$ is a rectangular matrix of low rank (known as eigenchannel matrix), $\mathbf{x}$ is a vector distributed with standard normal distribution. This is equivalent to saying that $\mathbf{c}$ is normally distributed with zero mean and covariance $\mathbf{U}\mathbf{U}^*$. The components of $\mathbf{x}$ are the channel factors in factor analysis modeling.

The underlying task in JFA is to train the hyperparameters $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{D}$ on a large training set. In the Bayesian framework, posterior distribution of the factors (knowing their priors) can be computed using the enrollment data. The likelihood of test utterance $\mathcal{X}$ is then computed by integrating over the posterior distribution of $\mathbf{y}$ and $\mathbf{z}$, and the prior distribution of $\mathbf{x}$ [Kenny and Dumouchel, 2004]. In [Kenny et al., 2005], it was later shown, that using mere MAP point estimates of $\mathbf{y}$ and $\mathbf{z}$ is sufficient. Still, integration over the prior distribution of $\mathbf{x}$ was performed. We will further show, that using the MAP point estimate of $\mathbf{x}$ gives comparable results. Scoring is understood as computing the log-likelihood ratio (LLR) between the target speaker model $\mathbf{s}$ and the UBM, for the test utterance $\mathcal{X}$.

There are many ways in which JFA can be trained and which different sites have experimented with. Not only the training algorithms differ, but also the results were reported using different scoring strategies.

2.1.1 Frame by Frame

Frame-by-Frame is based on a full GMM log-likelihood evaluation. The log-likelihood of utterance $\mathcal{X}$ and model $\mathbf{s}$ is computed as an average frame log-likelihood [1]. It is practically infeasible to integrate out the channel, therefore MAP point estimate of $\mathbf{x}$ is used. The formula is as follows

$$\log P(\mathcal{X}|\mathbf{s}) = \sum_{t=1}^{T} \log \sum_{c=1}^{C} w_c \mathcal{N}\left(\mathbf{o}_t; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right), \tag{2.4}$$

where $\mathbf{o}_t$ is the feature vector at frame t, T is the length (in frames) for utterance $\mathcal{X}$, C is number of Gaussians in the GMM, and w_c, $\boldsymbol{\Sigma}_c$, and $\boldsymbol{\mu}_c$ the cth Gaussian weight, mean, and covariance matrix, respectively.

2.1.2 Integrating over Channel Distribution

This approach is based on evaluating an objective function as given by Equation (13) in [Kenny et al., 2007]:

$$P(\mathcal{X}|\mathbf{s}) \quad = \quad \int P(\mathcal{X}|\mathbf{s}, \mathbf{x}) \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{I}) \mathrm{d}\mathbf{x} \tag{2.5}$$

As was said in the previous paragraph, it would be difficult to evaluate this formula in the frame-by-frame strategy. However, (2.4) can be approximated by using fixed alignment of frames to Gaussians, i.e., assume that each frame is generated by a single (best scoring) Gaussian. In this case, the likelihood can be evaluated in terms of the sufficient statistics. If the statistics are collected in the Baum-Welch way, the approximation is equal to the GMM EM auxiliary function, which is a lower bound to (2.5). The closed form (logarithmic) solution is then given

[1] All scores are normalized by frame length of the tested utterance, therefore the log-likelihood is average.

as:

$$\begin{aligned}\log \tilde{P}(\mathcal{X}|\mathbf{s}) &= \sum_{c=1}^{C} N_c \log \frac{1}{(2\pi)^{F/2}|\boldsymbol{\Sigma}_c|^{1/2}} \\ &\quad -\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_{\mathbf{s}}) - \frac{1}{2}\log|\mathbf{L}| \\ &\quad +\frac{1}{2}\|\mathbf{L}^{-1/2}\mathbf{U}^{*}\boldsymbol{\Sigma}^{-1}\mathbf{F}_{\mathbf{s}}\|^{2} \qquad (2.6)\end{aligned}$$

where for the first term, C is the number of Gaussians, N_c is the data count for Gaussian c, F is the feature vector size, $\boldsymbol{\Sigma}_c$ is covariance matrix for Gaussian c. These numbers will be equal both for UBM and the target model, thus the whole term will cancel out in the computation of the log-likelihood ratio.

For the second term of (2.6), $\boldsymbol{\Sigma}$ is the block-diagonal matrix of separate covariance matrices for each Gaussian, $\mathbf{S}_{\mathbf{s}}$ is the second order moment of $\mathcal{X}$ around speaker $\mathbf{s}$ given as

$$\mathbf{S}_{\mathbf{s}} = \mathbf{S} - 2\mathrm{diag}(\mathbf{F}\mathbf{s}^{*}) + \mathrm{diag}(\mathbf{N}\mathbf{s}\mathbf{s}^{*}), \qquad (2.7)$$

where $\mathbf{S}$ is the $CF \times CF$ block-diagonal matrix whose diagonal blocks are uncentered second order cumulants $\mathbf{S}_c$. This term is independent of speaker, thus will cancel out in the LLR computation (note that this was the only place where second order statistics appeared, therefore are not needed for scoring). $\mathbf{F}$ is a $CF \times 1$ vector, obtained by concatenating the first order statistics. $\mathbf{N}$ is a $CF \times CF$ diagonal matrix, whose diagonal blocks are $N_c\mathbf{I}_F$, i.e., the occupation counts for each Gaussian ($\mathbf{I}_F$ is $F \times F$ identity matrix).

The $\mathbf{L}$ in the third term of (2.6) is given as

$$\mathbf{L} = \mathbf{I} + \mathbf{U}^{*}\boldsymbol{\Sigma}^{-1}\mathbf{N}\mathbf{U}, \qquad (2.8)$$

where $\mathbf{I}$ is a $CF \times CF$ identity matrix, $\mathbf{U}$ is the eigenchannel matrix, and the rest is as in the second term. The whole term, however, does not depend on speaker and will cancel out in the LLR computation.

In the fourth term of (2.6), let $\mathbf{L}^{1/2}$ be a lower triangular matrix, such that

$$\mathbf{L} = \mathbf{L}^{1/2}\mathbf{L}^{1/2*} \qquad (2.9)$$

i.e., $\mathbf{L}^{-1/2}$ is the inverse of the Cholesky decomposition of $\mathbf{L}$.

As was said, terms one and three in (2.6), and second order statistics $\mathbf{S}$ in (2.7) will cancel out. Then the formula for the score is given as

$$\begin{aligned}Q_{\mathrm{int}}(\mathcal{X}|\mathbf{s}) &= \mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathrm{diag}(\mathbf{F}\mathbf{s}^{*})) \\ &\quad +\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathrm{diag}(\mathbf{N}\mathbf{s}\mathbf{s}^{*})) \\ &\quad +\frac{1}{2}\|\mathbf{L}^{-1/2}\mathbf{U}^{*}\boldsymbol{\Sigma}^{-1}\mathbf{F}_{\mathbf{s}}\|^{2} \qquad (2.10)\end{aligned}$$

2.1.3 Channel Point Estimate

This function is similar to the previous case, except for the fact, that the channel factor $\mathbf{x}$ is known. This way, there is no need for integrating over the whole distribution of $\mathbf{x}$, and only its point estimate is taken for LLR computation. The formula is directly adopted from [Kenny, 2005] (Theorem 1),

$$\begin{aligned}\log \tilde{P}(\mathcal{X}|\mathbf{s},\mathbf{x}) &= \sum_{c=1}^{C} N_c \log \frac{1}{(2\pi)^{F/2}|\mathbf{\Sigma}_c|^{1/2}} \\ &\quad -\frac{1}{2}\mathrm{tr}(\mathbf{\Sigma}^{-1}\mathbf{S}) \\ &\quad +\mathbf{M}^*\mathbf{\Sigma}^{-1}\mathbf{F} + \frac{1}{2}\mathbf{M}^*\mathbf{N}\mathbf{\Sigma}^{-1}\mathbf{M}, \end{aligned} \tag{2.11}$$

where $\mathbf{M}$ is given by (2.1). In this formula, the first and second terms cancel out in LLR computation, leading to scoring function

$$\begin{aligned}Q_{\mathrm{x}}(\mathcal{X}|\mathbf{s},\mathbf{x}) &= \mathbf{M}^*\mathbf{\Sigma}^{-1}\mathbf{F} \\ &\quad +\frac{1}{2}\mathbf{M}^*\mathbf{N}\mathbf{\Sigma}^{-1}\mathbf{M}, \end{aligned} \tag{2.12}$$

hence

$$\mathrm{LLR}_{\mathrm{x}}(\mathcal{X}|\mathbf{s}) = Q_{\mathrm{x}}(\mathcal{X}|\mathbf{s},\mathbf{x}_{\mathbf{s}}) - Q_{\mathrm{x}}(\mathcal{X}|\mathrm{UBM},\mathbf{x}_{\mathrm{UBM}}), \tag{2.13}$$

where $\mathbf{x}_{\mathrm{UBM}}$ is a channel factor estimated using UBM, and $\mathbf{x}_{\mathbf{s}}$ is a channel factor estimated using speaker $\mathbf{s}$.

2.1.4 UBM Channel Point Estimate

In [Vair et al., 2007], the authors assumed, that the shift of the model caused by the channel is identical both to the target model and the UBM[2]. Therefore, the $\mathbf{x}$ factor for utterance $\mathcal{X}$ is estimated using the UBM and then used for scoring. Formally written:

$$\begin{aligned}\mathrm{LLR}_{\mathrm{LPT}}(\mathcal{X}|\mathbf{s}) &= Q_{\mathrm{x}}(\mathcal{X}|\mathbf{s},\mathbf{x}_{\mathrm{UBM}}) \\ &\quad -Q_{\mathrm{x}}(\mathcal{X}|\mathrm{UBM},\mathbf{x}_{\mathrm{UBM}}) \end{aligned} \tag{2.14}$$

Note, that when computing the LLR, the $\mathbf{Ux}$ in the linear term of (2.11) will cancel out, leaving the compensation to the quadratic term of (2.11).

2.1.5 Linear Scoring

Let us keep the LPT assumption and let $\mathbf{m_c}$ be the channel compensated UBM:

$$\mathbf{m_c} = \mathbf{m} + \mathbf{c}. \tag{2.15}$$

Furthermore, let us assume, that we move the origin of supervector space to $\mathbf{m_c}$.

$$\bar{\mathbf{M}} = \mathbf{M} - \mathbf{m_c} \tag{2.16}$$

$$\bar{\mathbf{F}} = \mathbf{F} - \mathbf{N}\mathbf{m_c}. \tag{2.17}$$

Eq. (2.12) can now be rewritten to

$$\begin{aligned}Q_{\mathrm{xmod}}(\mathcal{X}|\bar{\mathbf{M}},\mathbf{x}) &= \bar{\mathbf{M}}^*\mathbf{\Sigma}^{-1}\bar{\mathbf{F}} \\ &\quad +\frac{1}{2}\bar{\mathbf{M}}^*\mathbf{N}\mathbf{\Sigma}^{-1}\bar{\mathbf{M}}. \end{aligned} \tag{2.18}$$

[2] The authors identified themselves under abbreviation LPT, therefore we will refer to this approach as to LPT assumption

When approximating (2.18) by the first order Taylor series (as a function of $\bar{\mathbf{M}}$), only the linear term is kept, leading to

$$Q_{\text{lin}}(\mathcal{X}|\bar{\mathbf{M}}, \mathbf{x}) = \bar{\mathbf{M}}^{*}\mathbf{\Sigma}^{-1}\bar{\mathbf{F}} \tag{2.19}$$

Realizing, that the channel compensated UBM is now a vector of zeros, and substituting (2.19) to (2.14), the formula for computing the LLR simplifies to

$$\text{LLR}_{\text{lin}}(\mathcal{X}|\mathbf{s}, \mathbf{x}) = (\mathbf{V}\mathbf{y} + \mathbf{D}\mathbf{z})^{*}\mathbf{\Sigma}^{-1}(\mathbf{F} - \mathbf{N}\mathbf{m} - \mathbf{N}\mathbf{c}). \tag{2.20}$$

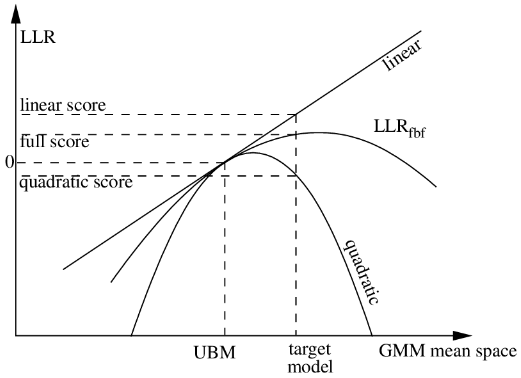


Figure 2.1: An illustration of the scoring behavior for frame-by-frame, LPT, and linear scoring.

Given the fact, that the $\tilde{P}$-function is a lower bound approximation of the real frame-by-frame likelihood function, there are cases, when the LPT original function fails. Fig. 2.1 shows that the linear function can sometimes be a better approximation of the full LLR.

2.2 Experimental setup

2.2.1 Test Set

The results of our experiments are reported on the Det1 and Det3 conditions of the NIST 2006 speaker recognition evaluation (SRE) dataset [NIST, ndb].

The real-time factor was measured on a special test set, where 49 speakers were tested against 50 utterances. The speaker models were taken from the t-norm cohort, while the test utterances were chosen from the original z-norm cohort, each having approximately 4 minutes, totally giving 105 minutes.

2.2.2 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 are calculated every 10 ms. This 20-dimensional feature vector was subjected to feature warping [Pelecanos and Sridharan, 2006] using a 3 s sliding window. Delta and double delta coefficients were then calculated using a 5 frames window giving a 60-dimensional feature vectors. These feature vectors were modeled using GMM and factor analysis was used to treat the problem of speaker and session variability.

Segmentation was based on the BUT Hungarian phoneme recognizer [Schwarz et al., 2006] and relative average energy thresholding. Also short segments were pruned out, after which the speech segments were merged together.

2.2.3 JFA Training

We used gender independent Universal Background Models, which contain 2048 Gaussians. This UBM was trained using LDC releases of Switchboard II, Phases 2 and 3; switchboard Cellular, Parts 1 and 2 and NIST 2004-2005 SRE. The (gender independent) factor analysis models were trained on the same quantities of data as the UBM.

Our JFA is composed by 300 speaker factors, 100 channel factors, and diagonal matrix **D**. While **U** was trained on the NIST data olny, **D** and **V** were trained on two disjoint sets comprising NIST and Switchboard data.

2.2.4 Normalization

All scores, as presented in the previous sections, were normalized by the number of frames in the test utterance. In case of normalizing the scores (zt-norm), we worked in the gender dependent fashion. We used 220 female, and 148 male speakers for t-norm, and 200 female, 159 male speakers for z-norm. These segments were a subset of the JFA training data set.

2.2.5 Hardware and Software

The frame-by-frame scoring was implemented in C++ code, which calls ATLAS functions for math operations. Matlab was used for the rest of the computations. Even though C++ produces more optimized code, the most CPU demanding computations are performed via the tuned math libraries that both Matlab and C++ use. This fact is important for measuring the real-time factor. The machine on which the real-time factor (RTF) was measured was a Dual-Core AMD Opteron 2220 with cache size 1024 KB. For the rest of the experiments, computing cluster was used.

2.3 Results

Table 2.1 shows the results without any score normalization. The reason for the loss of performance in the case of LPT scoring could possibly be due to bad approximation of the likelihood function around UBM, ,i.e., the inability to adapt the model to the test utterance (in the **U** space only). Fig. 2.1 shows this case.

Table 2.1: *Comparison of different scoring techniques in terms of EER and DCF. No score normalization was performed here.*

	Det1		Det3	
	EER	DCF	EER	DCF
Frame-by-Frame	**4.70**	**2.24**	**3.62**	**1.76**
Integration	5.36	2.46	4.17	1.95
Point estimate	5.25	2.46	4.17	1.96
Point estimate LPT	16.70	6.84	15.05	6.52
Linear	5.53	2.97	3.94	2.35

Table 2.2 shows the results after application of zt-norming. While the frame-by-frame scoring outperformed all the fast scorings in the un-normalized case, normalization is essential for the other methods.

Table 2.2: *Comparison of different scoring techniques in terms of EER and DCF. zt-norm was used as score normalization.*

	Det1		Det3	
	EER	DCF	EER	DCF
Frame-by-Frame	2.96	1.50	1.80	0.91
Integration	2.90	1.48	1.78	0.91
Point estimate	**2.90**	**1.47**	1.83	**0.89**
Point estimate LPT	3.98	2.01	2.70	1.36
Linear	2.99	1.48	**1.73**	0.95

2.3.1 Speed

The aim of this experiment was to show the approximate real time factor of each of the systems. The time measured included reading necessary data connected with the test utterance (features, statistics), estimating the channel shifts, and computing the likelihood ratio. Any other time, such as reading of hyper-parameters, models, etc. was not comprised in the result. Each measuring was repeated 5 times and averaged. Table 2.3 shows the real time of each algorithm. Surprisingly, the integration LLR is faster then the point estimate. This is due to implementation, where the channel compensation term in the integration formula is computed once per an utterance, while in the point estimate case, each model needs to be compensated for each trial utterance.

Table 2.3: *Real time factor for different systems*

	Time [s]	RTF
Frame-by-Frame	1010	1.60e^{-1}
Integration	50	7.93e^{-3}
Point estimate	160	2.54e^{-2}
Point estimate LPT	36	5.71e^{-3}
Linear	**13**	2.07e^{-3}

2.4 Conclusions

We have showed a comparison of different scoring techniques that different sites have recently used in their evaluations. While, in most cases, the performance does not change dramatically, the speed of evaluation is the major difference. The fastest scoring method is the Linear scoring. It can be implemented by a simple dot product, allowing for fast scoring of huge problems (e.g., z-, t- norming).

Chapter 3

i-vectors

The i-vector systems have become the state-of-the-art technique in the speaker verification field [Dehak et al., 2010]. 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.

The history of i-vectors is dated to summer 2008 JHU workshop on Robust Speaker Recognition [Burget et al., 2008]. At that time, JFA was the state-of-the-art technique and it was the centerpoint of interest among the workshop researchers. One of the directions was to use JFA as feature extraction. Various experiments were carried out on the JFA factors; SVM classification was studied, and different measures were tested to substitute the (fairly complicated) SVMs. There was an unofficial internal competition between the SVM and the dot-product sub-teams which was usually reflected in building touch-rugby or frisbee teams. Nevertheless, both teams found that using the channel factors for speaker detection gives around 20% EER and when fusing with the speaker factors, noticeable improvement was gained. Najim Dehak then came up with the idea of reducing the complexity of JFA to having only one multivariate hidden variable that would carry the total-variability information. He has originally called it the t-vector as for “total”, but the community quickly adopted the term i-vectors as for “intermediate”, “intervening”, “intelligent”, “informative”, “identity”, etc.

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.

3.1 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:

$$\mathbf{s} = \mathbf{m} + \mathbf{T}\mathbf{w} \tag{3.1}$$

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 $\mathbf{w}$:

$$\mathrm{p}(\mathbf{w}|\mathcal{X}) = \mathcal{N}(\mathbf{w}; \mathbf{w}_\mathcal{X}, \mathbf{L}_\mathcal{X}^{-1}) \tag{3.2}$$

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.

3.1.1 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:

$$N_\mathcal{X}^{(c)} = \sum_t \gamma_t^{(c)} \tag{3.3}$$

$$\mathbf{f}_\mathcal{X}^{(c)} = \sum_t \gamma_t^{(c)} \mathbf{o}_t \tag{3.4}$$

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 $\mathbf{f}_\mathcal{X} = \left(\mathbf{f}_\mathcal{X}^{(1)\prime}, \ldots, \mathbf{f}_\mathcal{X}^{(C)\prime}\right)'$, and $\mathbf{n}_\mathcal{X} = \left(N_\mathcal{X}^{(1)}, \ldots, N_\mathcal{X}^{(C)}\right)'$.

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]:

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

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 $\mathbf{T} = \left(\mathbf{T}^{(1)\prime}, \ldots, \mathbf{T}^{(C)\prime}\right)'$.

3.1.2 Parameter Estimation

As described in [Kenny, 2005] and with the data transforms from previous section, for an observation $\mathcal{X}$, the corresponding i-vector is computed as a point estimate:

$$\mathbf{w}_\mathcal{X} = \mathbf{L}_\mathcal{X}^{-1} \mathbf{T}' \mathbf{f}_\mathcal{X} \tag{3.5}$$

[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 [Kenny, 2005] for detailed formulae.

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

$$\mathbf{L}_{\mathcal{X}} = \mathbf{I} + \sum_{c=1}^{C} N_{\mathcal{X}}^{(c)} \mathbf{T}^{(c)\prime} \mathbf{T}^{(c)} \tag{3.6}$$

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 (3.6) 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 (3.5), and the second term represents the pre-computed matrices in the sum of (3.6).

Note that the computation complexity grows quadratically with M in the sum of (3.6), and linearly with C. This becomes the bottle-neck in the i-vector computation, resulting in high memory and CPU demands.

3.1.3 Model Training

Model hyper-parameters $\mathbf{T}$ are estimated using the same EM algorithm as in case of JFA [Kenny, 2005]. Note that our algorithm makes use of an additional *minimum divergence* update step [Kenny et al., 2007, Brümmer, 2009], which yields a quicker convergence, but is not described here.

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

$$\mathbf{C} = \sum_{i} \mathbf{f}_i \mathbf{w}_i' \tag{3.7}$$

$$\mathbf{A}^{(c)} = \sum_{i} N_i^{(c)} \left(\mathbf{L}_i^{-1} + \mathbf{w}_i \mathbf{w}_i' \right) \tag{3.8}$$

where $\mathbf{w}_i$ and $\mathbf{L}_i$ are the estimates from (3.5) and (3.6) for observation i. The M step update is given as follows:

$$\mathbf{T}^{(c)} = \mathbf{C}\mathbf{A}^{(c)^{-1}} \tag{3.9}$$

3.2 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 (3.3) are replaced by their prior probabilities represented by the UBM GMM weights. The new zero-order statistics are then:

$$\bar{N}_{\mathcal{X}}^{(c)} = \omega^{(c)} N_{\mathcal{X}} \tag{3.10}$$

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 (3.6) by $\bar{N}_{\mathcal{X}}^{(c)}$ from (3.10), we get

$$\bar{\mathbf{L}}_{\mathcal{X}} = \mathbf{I} + N_{\mathcal{X}} \mathbf{W} \tag{3.11}$$

where

$$\mathbf{W} = \sum_{c=1}^{C} \omega^{(c)} \mathbf{T}^{(c)\prime} \mathbf{T}^{(c)} \tag{3.12}$$

Exploiting this simplification in the i-vector extractor training can be done at two stages: substituting $\mathbf{L}_i$ in (3.8) by (3.11), and substituting $N_i^{(c)}$ in (3.8) by (3.10). 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 (3.12) 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 $O(CFM + M^2)$.

3.3 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 (3.6) as

$$\mathbf{L}_{\mathcal{X}} = \mathbf{G}^{(-1)'}\hat{\mathbf{L}}_{\mathcal{X}}\mathbf{G}^{-1} \tag{3.13}$$

where

$$\hat{\mathbf{L}}_{\mathcal{X}} = \mathbf{G}'\mathbf{G} + \sum_{c=1}^{C} N_{\mathcal{X}}^{(c)}\mathbf{G}'\mathbf{T}^{(c)'}\mathbf{T}^{(c)}\mathbf{G} \tag{3.14}$$

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

$$\hat{\mathbf{L}}_{\mathcal{X}} = \operatorname{Diag}\left(\operatorname{diag}(\mathbf{G}'\mathbf{G}) + \mathbf{V}\mathbf{n}_{\mathcal{X}}\right) \tag{3.15}$$

where $\mathbf{V}$ is a $M \times C$ matrix whose cth column is $\operatorname{diag}(\mathbf{G}'\mathbf{T}^{(c)'}\mathbf{T}^{(c)}\mathbf{G})$. $\operatorname{Diag}(\cdot)$ maps a vector to a diagonal matrix, while $\operatorname{diag}(\cdot)$ maps a matrix diagonal to a vector. Combining (3.13) and (3.5), we get

$$\hat{\mathbf{w}}_{\mathcal{X}} = \mathbf{G}\hat{\mathbf{L}}_{\mathcal{X}}^{-1}\mathbf{G}'\mathbf{T}'\mathbf{f}_{\mathcal{X}} \tag{3.16}$$

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 (3.15).

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

3.3.1 Eigen-decomposition

Let $\mathbf{W}$ be the weighted average per-component covariance matrix from (3.12). We assume $\mathbf{W}$ to be a full-rank matrix with M linearly independent eigenvectors. Then $\mathbf{W}$ can be factorized as

$$\mathbf{W} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{-1} \tag{3.17}$$

where $\mathbf{Q}$ is a square $M \times M$ matrix whose ith column is the eigenvector $\mathbf{q}_i$ of $\mathbf{W}$ and $\boldsymbol{\Lambda}$ 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 $\mathbf{G} = \mathbf{Q}$.

3.3.2 Heteroscedastic Linear Discriminant Analysis

If the average covariance matrix $\mathbf{W}$ from (3.12) 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 decorrelates features associated with each particular class (maximum likelihood linear transformation for diagonal covariance modeling [Kumar, 1997]). An efficient iterative algorithm [Gales, 1999] 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 $\mathbf{T}^{(c)'}\mathbf{T}^{(c)}$, 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.

3.4 Experimental setup

3.4.1 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.

3.4.2 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.

3.4.3 Scoring and Normalization

The same technique as in [Dehak et al., 2010] 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:

$$\text{score}\left(\mathbf{w}_{\text{target}}, \mathbf{w}_{\text{test}}\right) = \frac{\langle \mathbf{w}_{\text{target}}, \mathbf{w}_{\text{test}} \rangle}{\|\mathbf{w}_{\text{target}}\| \|\mathbf{w}_{\text{test}}\|} \tag{3.18}$$

The cosine distance scores were normalized using gender-dependent s-norm [Brümmer and Strasheim, 2009] with a cohort of 400 speakers having 2 utterances per speaker.

3.4.4 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 [NIST, ndb]. 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 (3.6)
- $\mathbf{W}$ in (3.12)
- $\mathbf{G}$ and $\mathbf{T}^{(c)}\mathbf{G}$ in (3.13) and (3.16), and $\mathbf{V}$ in (3.15)

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.

3.5 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. 3.1.2, and *simple 1* and *simple 2* reference to the proposed simplifications.

Table 3.1 summarizes the systems with respect to verification accuracy. Fig. 3.1 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”.

3.5.1 Speed and Memory

As described earlier in Sec. 3.4.4, the computation time does not include reading of the necessary data and pre-computation of some terms. The results are reported in Tab. 3.2. The dominating complexity of matrix inversion makes “simple 2” faster than “simple 1”, as described in Sec. 3.2 and 3.3.

Tab. 3.3 shows memory allocation for different systems. We see that for most of the current hardware configurations, the baseline systems could be a problem.

Note that prior to the scoring, WCCN and LDA dimensionality reduction are applied to the i-vectors (see Sec. 3.4.3). Projecting this linear transformation directly into the leftmost $\mathbf{G}$ of (3.16) could further decrease the complexity of the “simple 2” algorithm.

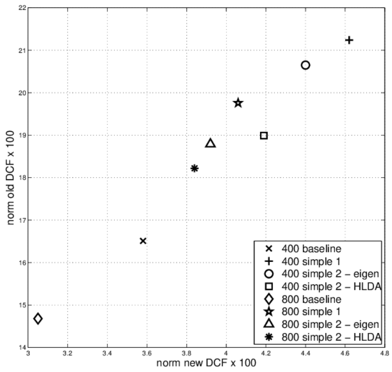


Figure 3.1: Constellation plot of the individual systems

Table 3.1: *Comparison of the proposed i-vector extraction methods in terms of normalized DCFs and EER*

	DCF_{new}	DCF_{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

3.5.2 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 3.4 shows the difference.

Time and memory complexity of collecting the accumulators $\mathbf{A}$ from (3.8) is almost identical to the computation of $\mathbf{L}_{\mathcal{X}}$ in (3.6). The proposed method still keeps the same accumulator collection, however, avoiding the expensive computation of (3.6) decreases the E step time and memory complexity by a factor of 2.

3.5.3 The MOBIO Experiments

This experiment shows the methods with the scaled-down system that was used on the cell phone as the part of the MOBIO project [S. Marcel (IDIAP),].

The VAD is essentially the same as in the rest of the experiments except the Czech phoneme recognizer was used instead of Hungarian. The features were 20-dimensional MFCC's augmented with their first- and second-order derivatives. Short-time cepstral mean and variance normalization was applied over 3s windows.

The average length of each utterance for the MOBIO database is around 5 seconds. Therefore, we had to join all utterances from one speaker together for training the i-vector extractor.

Table 3.2: *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%

Table 3.3: *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

The extractor was gender-independent, 256G and 128G are tested, s-norm was applied. The train-set for the i-vector extractor and WCCN and LDA training was the same as in the previous experiments. Testing was performed on the original (short) utterances.

Table 3.5 shows the results. It is apparent that the difference between the methods is not as noticeable as in the NIST case. The reason is that the test segments are much shorter than the ones of NIST. This results in much broader posterior distributions of the i-vectors, i.e. the scale of $\mathbf{L}^{-1}$, which corresponds to overall performance degradation, after which, the methods start to perform similarly.

3.6 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 did we manage 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.

Table 3.4: *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

Table 3.5: *Comparison of the proposed i-vector extractor training methods in terms of normalized DCFs and EER for MOBIO*

	female		male	
	$\mathrm{DCF}_{\mathrm{old}}$	EER	$\mathrm{DCF}_{\mathrm{old}}$	EER
128G baseline	0.0632	13.55	0.0597	14.50
128G simple 1	0.0635	14.08	0.0607	15.19
256G baseline	0.0588	12.94	0.0599	14.36
256G simple 1	0.0580	12.29	0.0599	13.81

Chapter 4

Discriminative Training

Recently, systems based on i-vectors [Dehak et al., 2010, Kenny, 2010] (extracted from cepstral features) have provided superior performance in speaker verification. The so-called i-vector is an information-rich low-dimensional fixed-length vector extracted from the feature sequence representing a speech segment. A speaker verification score is produced by comparing two i-vectors corresponding to the segments in the verification trial. The function taking two i-vectors as an input and producing the corresponding verification score is designed to give the log-likelihood ratio between the "same-speaker" and "different-speaker" hypotheses. Best performance is currently obtained with Probabilistic Linear Discriminant Analysis (PLDA) [Kenny, 2010]—a generative model that models i-vector distributions allowing for direct evaluation of the desired log-likelihood ratio verification score (see Section 4.1.4 for details).

In [Burget et al., 2011], it was shown that discriminatively training the PLDA parameters can lead to improvement in recognition performance. In this paper, we go deeper in the speaker recognition chain and we show that a similar discriminative training framework can be adopted for training the parameters of the i-vector extractor. We apply this technique in two kinds of i-vector extractor. In the first case, the traditional extraction—as proposed in [Dehak et al., 2010]—is studied. It will be further referred to as the *full i-vector extractor*. Its parameters are given by a single matrix **T**. In the second case, the simplified extraction (referred to as "Simplification 2" in [Glembek et al., 2011b]) is addressed. Its parameters are given by three matrices—**T**, **G**, and **V**. It will be further referred to as the *simplified i-vector extractor*.

4.1 Theoretical background

The i-vectors provide an elegant way of reducing 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 (JFA) framework introduced in [Kenny, 2005, Kenny et al., 2007].

The main idea is that the speaker- and channel-dependent Gaussian Mixture Model (GMM) supervector **s** can be modeled as:

$$\mathbf{s} = \mathbf{m} + \mathbf{T}\mathbf{w} \tag{4.1}$$

where **m** is the Universal Background Model (UBM) GMM mean supervector, **T** is a low-rank matrix representing M bases spanning subspace with important variability in the mean supervector space, and **w** is a latent variable of size M with standard normal distribution.

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

$$\mathrm{p}(\mathbf{w}|\mathcal{X}) = \mathcal{N}(\mathbf{w}; \mathbf{w}_{\mathcal{X}}, \mathbf{L}_{\mathcal{X}}^{-1}) \tag{4.2}$$

The i-vector $\boldsymbol{\phi}$ is the Maximum a Posteriori (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. $\mathbf{L}_{\mathcal{X}}$ is the precision of the posterior distribution.

4.1.1 Sufficient statistics

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 $\boldsymbol{\Sigma}^{(c)}$, and a vector of mixture weights $\boldsymbol{\omega}$. For each Gaussian component c, the statistics are given respectively as

$$N_{\mathcal{X}}^{(c)} = \sum_t \gamma_t^{(c)} \tag{4.3}$$

$$\mathbf{f}_{\mathcal{X}}^{(c)} = \sum_t \gamma_t^{(c)} \mathbf{o}_t \tag{4.4}$$

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 $\mathbf{f}_{\mathcal{X}} = \left(\mathbf{f}_{\mathcal{X}}^{(1)\prime}, \ldots, \mathbf{f}_{\mathcal{X}}^{(C)\prime}\right)'$, and $\mathbf{n}_{\mathcal{X}} = \left(N_{\mathcal{X}}^{(1)}, \ldots, N_{\mathcal{X}}^{(C)}\right)'$.

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:

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

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

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

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

4.1.2 i-vector extraction

As described in [Kenny, 2005] and with the data transforms from the previous section, for an observation $\mathcal{X}$, the corresponding i-vector is computed as a point estimate:

$$\boldsymbol{\phi}_{\mathcal{X}} = \mathbf{L}_{\mathcal{X}}^{-1} \mathbf{T}' \mathbf{f}_{\mathcal{X}} \tag{4.5}$$

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

$$\mathbf{L}_{\mathcal{X}} = \mathbf{I} + \sum_{c=1}^{C} N_{\mathcal{X}}^{(c)} \mathbf{T}^{(c)\prime} \mathbf{T}^{(c)} \tag{4.6}$$

[1] Part of the factor computation is the evaluation of $\mathbf{T}'\boldsymbol{\Sigma}^{-1}\mathbf{f}$, where the decomposed $\boldsymbol{\Sigma}^{-1}$ can be projected to the neighboring terms, see [Kenny, 2005] for detailed formulae.

4.1.3 i-vector extraction—simplified version

According to [Glembek et al., 2011b], the i-vector extraction can be simplified to reduce the computation complexity. Assuming there is a linear (orthogonal) transformation $\mathbf{G}$ that would orthogonalize all individual per-component submatrices $\mathbf{T}^{(c)}$, the i-vector extraction can be expressed as

$$\hat{\boldsymbol{\phi}}_{\mathcal{X}} = \mathbf{G}\hat{\mathbf{L}}_{\mathcal{X}}^{-1}\mathbf{G}'\mathbf{T}'\mathbf{f}_{\mathcal{X}} \tag{4.7}$$

where

$$\hat{\mathbf{L}}_{\mathcal{X}} = \mathrm{Diag}\left(\mathbf{I} + \mathbf{V}\mathbf{n}_{\mathcal{X}}\right) \tag{4.8}$$

where $\mathbf{V}$ is an $M \times C$ matrix whose cth column is $\mathrm{diag}(\mathbf{G}'\mathbf{T}^{(c)\prime}\mathbf{T}^{(c)}\mathbf{G})$. $\mathrm{Diag}(\cdot)$ maps a vector to a diagonal matrix.

4.1.4 PLDA

To facilitate comparison of i-vectors in a verification trial, we use a Probabilistic Linear Discriminant Analysis (PLDA) model [Prince and Elder, 2007, Kenny, 2010]. It can be seen as a special case of JFA with a single Gaussian component. Given a pair of i-vectors, PLDA allows to compute the log-likelihood for the same-speaker hypothesis and for the different-speaker hypothesis. One can directly evaluate the log-likelihood ratio of the same-speaker and different-speaker trial using

$$\begin{aligned} s(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) &= \boldsymbol{\phi}_1^T \boldsymbol{\Lambda} \boldsymbol{\phi}_2 + \boldsymbol{\phi}_2^T \boldsymbol{\Lambda} \boldsymbol{\phi}_1 + \boldsymbol{\phi}_1^T \boldsymbol{\Gamma} \boldsymbol{\phi}_1 + \boldsymbol{\phi}_2^T \boldsymbol{\Gamma} \boldsymbol{\phi}_2 \\ &+ (\boldsymbol{\phi}_1 + \boldsymbol{\phi}_2)^T \mathbf{c} + k, \end{aligned} \tag{4.9}$$

where $\boldsymbol{\Lambda}$, $\boldsymbol{\Gamma}$, $\mathbf{c}$, k are derived from the parameters of PLDA as in [Burget et al., 2011].

4.1.5 i-vector length normalization

PLDA assumes that the input i-vectors are normally distributed. However, in earlier studies ([Kenny, 2010]), it has been shown that this assumption is not met.

Length normalization [Dehak et al., 2010, Garcia-Romero, 2011] of the i-vectors forces them to lie on a unity sphere, which brings them closer to the Gaussian distribution shell where most of the probability density mass is concentrated. The transformation is given as

$$\bar{\boldsymbol{\phi}} = \frac{\boldsymbol{\phi}}{\|\boldsymbol{\phi}\|} = \frac{\boldsymbol{\phi}}{\sqrt{\boldsymbol{\phi}'\boldsymbol{\phi}}} \tag{4.10}$$

4.2 Discriminative classifier

We describe how we train the i-vector extractor parameters $\boldsymbol{\theta}$ in order to discriminate between same-speaker and different-speaker trials, without having to explicitly model the distributions of i-vectors.

The set of training examples, which we continue referring to as training trials, comprises both different-speaker, and same-speaker trials. Let us use the coding scheme $t \in \{-1, 1\}$ to represent labels for the different-speaker, and same-speaker trials, respectively. Assigning each trial a log-likelihood ratio s and the correct label t, the log probability of recognizing the trial correctly can be expressed as

$$\log p(t|\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) = -\log(1 + \exp(-st)). \tag{4.11}$$

In the case of logistic regression, the objective function to be maximized is the log probability of correctly classifying all training examples, i.e., the sum of expressions (4.11) evaluated for all training trials. Equivalently, this can be expressed by minimizing the cross-entropy error function, which is a sum over all training trials

$$E(\boldsymbol{\theta}) = \sum_{n=1}^{N} \alpha_n E_{LR}(t_n s_n) + \frac{\lambda}{2} \|\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{ML}}\|^2, \tag{4.12}$$

where the logistic regression loss function

$$E_{LR}(ts) = \log(1 + \exp(-ts)) \tag{4.13}$$

is simply the negative log probability (4.11) of correctly recognizing a trial. We have also added the regularization term $\frac{\lambda}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{ML}}\|^2$, where λ is a constant controlling the trade-off between the error function and the regularizer, and $\boldsymbol{\theta}_{\mathrm{ML}}$ is the original maximum-likelihood estimate of the given parameter. This kind of regularization is similar to the sum-of-squares penalty; however, it controls the distance from the original parameters rather than the parameter range itself. This way, optimizing the error function fine tunes the already good parameters.

The coefficients α_n allow us to weight individual trials. Specifically, we use them to assign different weights to same-speaker and different-speaker trials. This allows us to select a particular operating point, around which we want to optimize the performance of our system without relying on the proportion of same- and different-speaker trials in the training set. The advantage of using the cross-entropy objective for training is that it reflects performance of the system over a wide range of operating points (around the selected point).

4.2.1 Gradient evaluation

In order to numerically optimize the parameters $\boldsymbol{\theta}$, we want to express the gradient of the error function

$$\nabla E(\boldsymbol{\theta}) = \sum_{n=1}^{N} \alpha_n \frac{\partial E_{LR}(t_n s_n)}{\partial \boldsymbol{\theta}} + \lambda(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{ML}}). \tag{4.14}$$

We see that the loss function $E_{LR}(t_n s_n)$ is not directly dependent on $\boldsymbol{\theta}$; therefore, the chain rule must be subsequently applied.

Let us start by deriving the loss function w.r.t. the direct parameters of E_{LR}

$$\frac{\partial E_{LR}}{\partial \boldsymbol{\theta}} = \frac{\partial E_{LR}}{\partial s} \frac{\partial s}{\partial \boldsymbol{\theta}} \tag{4.15}$$

The first r.h.s. fraction of (4.15) is defined as

$$\frac{\partial E_{LR}(ts)}{\partial s} = -t\sigma(-ts), \tag{4.16}$$

where $\sigma(\cdot)$ is the logistic function. Noting that the score s is a function of a length-normalized i-vector pair

$$s = s(\bar{\boldsymbol{\phi}}_1, \bar{\boldsymbol{\phi}}_2),$$

we get

$$\frac{\partial s_n}{\partial \boldsymbol{\theta}} = \frac{s(\bar{\boldsymbol{\phi}}_1, \bar{\boldsymbol{\phi}}_2)}{\partial \bar{\boldsymbol{\phi}}_1} \frac{\partial \bar{\boldsymbol{\phi}}_1}{\partial \boldsymbol{\theta}} + \frac{s(\bar{\boldsymbol{\phi}}_1, \bar{\boldsymbol{\phi}}_2)}{\partial \bar{\boldsymbol{\phi}}_2} \frac{\partial \bar{\boldsymbol{\phi}}_2}{\partial \boldsymbol{\theta}} \tag{4.17}$$

From (4.9), knowing that $\boldsymbol{\Lambda}$ and $\boldsymbol{\Gamma}$ are symmetrical, we can derive

$$\frac{s(\bar{\boldsymbol{\phi}}_1, \bar{\boldsymbol{\phi}}_2)}{\partial \bar{\boldsymbol{\phi}}_1} = 2\boldsymbol{\phi}_2'\boldsymbol{\Lambda} + 2\boldsymbol{\phi}_1'\boldsymbol{\Gamma} + \mathbf{c} \tag{4.18}$$

Note that the two sides of the trial can be swapped so that an analogous equation applies when deriving w.r.t. $\boldsymbol{\phi}_2$. Again, we apply the chain rule to derive through the length normalization:

$$\frac{\partial \bar{\boldsymbol{\phi}}}{\partial \boldsymbol{\theta}} = \frac{\partial \bar{\boldsymbol{\phi}}}{\partial \boldsymbol{\phi}} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{\theta}} \tag{4.19}$$

where

$$\frac{\partial \bar{\boldsymbol{\phi}}}{\partial \boldsymbol{\phi}} = \frac{1}{\|\boldsymbol{\phi}\|} \left(\mathbf{I} - (\bar{\boldsymbol{\phi}}\bar{\boldsymbol{\phi}}') \right). \tag{4.20}$$

We get the full derivatives by applying the chain fule for differentials. In the case of the full i-vector extractor, the derivative can be expressed a

$$\begin{aligned} \frac{\partial E(\mathbf{T})}{\partial \mathbf{T}} = \sum_{j=1}^{M} - & \left(\mathbf{L}_j^{-1} \frac{\partial E}{\partial \boldsymbol{\phi}_j'} \boldsymbol{\phi}_j' + \boldsymbol{\phi}_j \frac{\partial E}{\partial \boldsymbol{\phi}_j} \mathbf{L}_j^{-1} \right) \mathbf{T}'\mathbf{N}_j \\ & + \mathbf{L}_j^{-1} \frac{\partial E}{\partial \boldsymbol{\phi}_j} \mathbf{f}_j, \end{aligned} \tag{4.21}$$

where $\mathbf{N}_j$ is a diagonal matrix, whose entries are $(N_j^{(1)}, \cdots, N_j^{(1)}, N_j^{(2)}, \cdots, N_j^{(2)}, \cdots)$, where every $N_j^{(i)}$ of $\mathbf{n}_j$ is expanded to match the feature dimensionality. For the simplified i-vector extraction, the derivatives of the parameters are

$$\frac{\partial E(\mathbf{T})}{\partial \mathbf{T}} = \sum_{j=1}^{M} \mathbf{f}_j \frac{\partial E}{\partial \boldsymbol{\phi}_j} \mathbf{G}\hat{\mathbf{L}}_j^{-1}\mathbf{G}' \tag{4.22}$$

$$\frac{\partial E(\mathbf{G})}{\partial \mathbf{G}} = \sum_{j=1}^{M} \hat{\mathbf{L}}_j^{-1}\mathbf{G}' \left(\mathbf{T}'\mathbf{f}_j \frac{\partial E}{\partial \boldsymbol{\phi}_j} + \frac{\partial E}{\partial \boldsymbol{\phi}_j'} \mathbf{f}_j'\mathbf{T} \right) \tag{4.23}$$

$$\frac{\partial E(\mathbf{V})}{\partial \mathbf{V}} = \sum_{j=1}^{M} -\mathbf{n}_j \left(\frac{\partial E}{\partial \boldsymbol{\phi}_j} \mathbf{G}' \circ \mathbf{f}_j'\mathbf{T}\mathbf{G}\hat{\mathbf{L}}_j^{-2} \right) \tag{4.24}$$

where the $\circ$ stands for the Hadamard product.

4.2.2 Experimental Setup

Test setup

The results of our experiments are reported on the female part of Condition 5 of the NIST 2010 SRE dataset [NIST, nda]. The recognition accuracy is given in terms of equal error rate (EER), and the normalized DCF as defined in both NIST 2010 SRE (DCF$_{\text{new}}$) and the previous SRE evaluations (DCF$_{\text{old}}$).

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 Gaussianization [Pelecanos and Sridharan, 2006] using a 3 s sliding window. Delta and double delta coefficients were then calculated using a five-frame window giving a 60-dimensional feature vector.

Segmentation was based on the Brno University of Technology (BUT) Hungarian phoneme recognizer and relative average energy thresholding. Also, short segments were pruned out, after which the speech segments were merged.

System Setup

One gender-independent UBM was represented as a diagonal covariance, 64-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.

The initial i-vector extractor **T** was trained on the female portion 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 speakers in 1463 hours of speech. The dimensionality of the i-vectors was set to 400. The initial orthogonalization matrix **G** was computed using HLDA, as described in Section 3.3.2. Length normalization was applied after i-vector extraction.

PLDA was trained using the same data set as the **T** matrix. Only the Fisher portion was trimmed off, reducing the amount of data by approximately 50%. The across-class covariance matrix (eigen-voices) was of rank 90, and the within-class covariance matrix (eigen-channels) was full-rank.

The training dataset for the discriminative training was identical to the dataset of PLDA. The cross-entropy function was evaluated on the complete trial set, i.e., all training samples were scored against each other, giving 378387 same-speaker trials, and over 468 million different-speaker trials.

Numerical optimization

The numerical optimization of the parameters was performed in matlab using the optimization and differentiation tools in the BOSARIS Toolkit [Brümmer and de Villiers, 2010]. It uses the trust region Newton conjugate gradient method, as described in [Lin et al., 2008, Nocedal and Wright, 2006]. In addition to the first derivatives as given in Section 4.2.1, this method needs to evaluate the second order Hessian-vector product [Pearlmutter, 1994], which can be effectively computed via the 'complex step differentiation' [Shampine, 2007].

Different values for the regularization coefficient λ were tested. Good convergence and stability were observed when setting it to 0.2 for the full i-vector extractor parameters, and 0.8 for the simplified version. In the case of the simplified version, the matrices **G** and **T** were optimized subsequently. It was found, however, that even though optimizing **V** kept on decreasing the error function, it would always decrease the recognition performance on the test set. Different regularizers were also tested; however, it turned out that together with good initialization, the discriminative training works only as a "fine-tuner" of the initial parameters. Target prior $p(\mathcal{H}_1)$ was set to 0.001 accroding to NIST2010 requirement.

Table 4.1: *Comparison of ML and discriminatively trained full i-vector extractors in terms of normalized DCFs and EER*

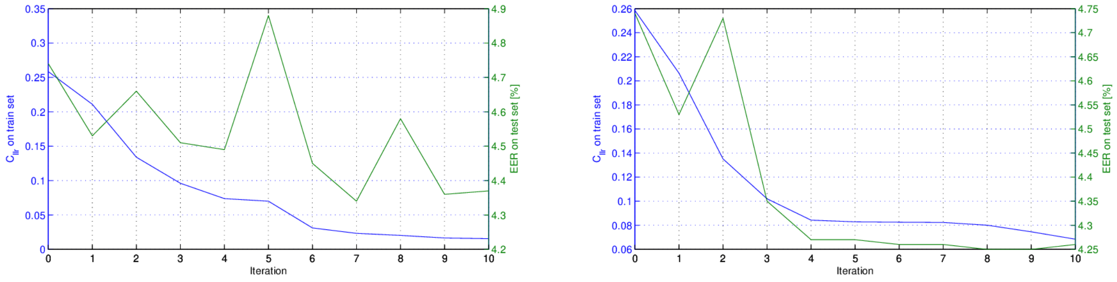
	DCF_{new}	DCF_{old}	EER
ML	0.6678	0.2200	4.74
discriminative	0.6548	0.2122	4.26

Table 4.2: *Comparison of ML and discriminatively trained simplified i-vector extractors in terms of normalized DCFs and EER*

	DCF_{new}	DCF_{old}	EER
ML	0.7496	0.2710	6.18
discriminative	0.6691	0.2403	5.41

Results

Table 4.1 shows the situation when training the full i-vector extractor. There is only a slight improvement in performance. In the case of the simplified i-vector extractor, the improvement is more apparent—see Table 4.2 for results. We see that the simplified system is still worse than the full one; however, discriminative training has shown its potential.



(a) No regularization

(b) With regularization, $\lambda = 0.2$

Figure 4.1: Plot of the objecitve function (C_{llr}) and the EER w.r.t. number of optimization iterations in discriminative training of the full ivector extractor. We can see the effect of regularizing the optimization—after the third iteration the EER is monotonically decreasing.

Chapter 5

Conclusions

5.1 Summary

In the first part of my work, I present comparison of different methods for scoring test utterances using the Joint Factor Analysis models. The methods differ in how they deal with the channel of the tested utterance. The work was inspired by the fact that many sites used JFA in slightly different ways and comparison of the results among sites was influenced not only by the methodology of training their systems, but also by the scoring procedures.

The first method is based on evaluating the real log-likelihood functions (frame-by-frame) for both the UBM and for the probed model—for each of the two, the channel factors are estimated separately and dealt with as point estimates—and the log-likelihood ratio is computed. The rest of the methods are based on approximating the likelihood functions using the fixed-alignment assumption, which allows for using the zero- and first-order statistics and simplifies the log-likelihood computation to a quadratic function of the model. It was shown that using the point estimates—as in the frame-by-frame case— with the fixed-alignment approximation and applying zt-norm gives results almost identical to the frame-by-frame approach, which confirms that the fixed-alignment is a good assumption. It was also shown that integrating the quadratic function over the posterior distribution of the channel factors gives results comparable to the point-estimate approach. This is due to the fact that the posterior distribution is sharp on long-enough utterances—in our case, the approximate length of the utterances was 2.5 minutes, which makes the point estimate a good approximation of the real posterior distribution. Estimating the channel point-estimate using the UBM only, and applying it to both likelihood functions of the likelihood ratio has shown to lead to worse results when used with the quadratic scoring. However, omitting the quadratic term—which can be interpreted as first-order Taylor series approximation—leads to further simplification of the log-likelihood function. Not only the computation of the whole log-likelihood ratio reduces to a simple dot-product function, but also the accuracy of the system is comparable to frame-by-frame approach. This method also allows for fast scoring of large-scale evaluation sets, especially when all-against-all scoring is needed. For experimental usage, linear scoring was found to be approximately 80 times faster than the frame-by-frame approach.

In the second part, the extraction of i-vectors was studied. I have proposed two simplifications to the i-vector computation. Both methods are based on approximating the covariance of the posterior distribution of the i-vector. The first method approximates the zero-order statistics by mere scaling of the GMM weights by the number of data-points. This way, I 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, I managed to reduce the complexity of the algorithm with sacrificing recognition accuracy by 20–30%, which makes this technique usable in small-scale devices. It was shown that for short utterances (in average 5 sec), the method performs similarly as the standard i-vector extraction. The results, with the equal error rates in the range of 12%, however, are dramatically worse than when using long utterances (which are typical in the NIST evaluations). The posterior distributions in both methods are very broad which makes the point estimates of the i-vector almost equally uncertain. As a practical result, Simplification 1 was used in the MOBIO project, when porting a speaker verification system on a mobile phone platform.

The second simplification is based on orthogonalization of the subspace and assuming that the posterior covariance is diagonal. Compared to the previous simplification, this approach leads to better performance both in terms of speed and accuracy. The degradation of accuracy for this technique, compared to the standard i-vector extraction, is around 17% on EER on the NIST2010 data.

In the third part, discriminative training in automatic speaker recognition was studied and adapted for the i-vector system. The objective for the training, as used in this work, is the cross-entropy. The work follows on previous experiments where the same objective was used for training the eigen-voices matrix of JFA, and later for training the parameters of PLDA. I have applied the technique both to the original i-vector extractor and to its simplified version, where orthogonal subspace is assumed. In both cases, the discriminative training was effective: 10% relative improvement was achieved for the standard i-vector extraction and 15% relative in the simplified case. The optimization was performed numerically and it it was found out that mere discriminative training does not work by itself. Rather, good initialization has to be provided—in our case the standard ML estimat. Discriminative training is then used to “fine-tune” the parameters.

5.2 Future Work

5.2.1 Low-hanging Fruit

Most of the ideas for future work are inspired by the last part of my work, i.e., the discriminative training. In this work, I summarized discriminative training of PLDA and I have described discriminative training of the i-vector extractor. However, for the later one, I have always used generatively trained PLDA. In this sense, the first thing that might be worth experimenting with is joint discriminative training of multiple parts of the speaker recognition system.

In my i-vector extractor discriminative training, I have always built the training set as a complete list of all possible trials, i.e., all-against-all strategy. It would be interesting to try to experiment with different trial sets. Another interesting experiment would be to cut the training utterances into large number of shorter segments. This thought is inspired by experiments in other fields of speech processing, such as language recognition [Matějka et al., 2006], where MMI technique started to be successful only when using short segments.

5.2.2 Long-term Plans

Concerning i-vector extraction with PLDA backend, it would also be interesting to discriminatively optimize the i-vector extraction while concerning simultaneous ML estimation of the PLDA parameters. This would make the parameters of the PLDA dependent on the i-vector extractor and the discriminative objective function dependent also on PLDA parameters, which would be ML-updated based on the changing i-vectors (where i-vectors depend on changing the parameters of i-vector extractor). This corresponds to an additional indirect dependence of the objective function on the i-vector extractor parameters, which has to be taken into account when evaluating gradient of the objective function w.r.t. the i-vector extractor parameters. This problem is similar to the one in discriminatively trained feature extraction used in ASR, namely fMPE [Povey et al., 2005].

As for the long-term plans, I am very interested in using subspace modeling in combination with other distributions. I have already experimented with channel compensation of the multinomial distribution [Glembek et al., 2008] and it has been shown in [Kockmann, 2012, Soufifar et al., 2011, D'Haro et al., 2012] that similar approach can be used for i-vector-like extraction for discrete data.

Bibliography

[Auckenthaler et al., 2000] Auckenthaler, R., Carey, M., and Lloyd-Thomas, H. (2000). Score normalization for text-independent speaker verification systems. *Digital Signal Processing*, 10(1-3):42–54.

[Brümmer, 2009] Brümmer, N. (2009). The EM algorithm and minimum divergence. Agnitio Labs Technical Report. Online: http://niko.brummer.googlepages.com/EMandMINDIV.pdf.

[Brümmer et al., 2007] Brümmer, N., Burget, L., Černocký, J., Glembek, O., Grézl, F., Karafiát, M., van Leeuwen, D., Matějka, P., Schwarz, P., and Strasheim, A. (2007). Fusion of heterogeneous speaker recognition systems in the STBU submission for the NIST speaker recognition evaluation 2006. *IEEE Transactions on Audio, Speech and Language Processing*, 15(7):2072–2084.

[Brümmer and de Villiers, 2010] Brümmer, N. and de Villiers, E. (2010). The BOSARIS toolkit. http://sites.google.com/site/bosaristoolkit/.

[Brümmer and Strasheim, 2009] Brümmer, N. and Strasheim, A. (2009). AGNITIO's speaker recognition system for EVALITA 2009.

[Burget et al., 2008] Burget, L., Brummer, N., Reynolds, D., Kenny, P., Pelecanos, J., Vogt, R., Castaldo, F., Dehak, N., Dehak, R., Glembek, O., Karam, Z., Noecker, J. J., Na, Y. H., Costin, C. C., Hubeika, V., Kajarekar, S., Scheffer, N., and Černocký, J. (2008). Robust speaker recognition over varying channels. Technical report, Johns Hopkins University.

[Burget et al., 2011] Burget, L., Plchot, O., Cumani, S., Glembek, O., Matějka, P., and Brümmer, N. (2011). Discriminatively trained probabilistic linear discriminant analysis for speaker verification. In *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, CZ.

[Campbell et al., 2006] Campbell, W., Sturim, D., Reynolds, D., and Solomonoff, A. (2006). SVM based speaker verification using a GMM supervector kernel and nap variability compensation. In *Proceedings of ICASSP 2006*, volume 1, page I.

[Campbell, 2002] Campbell, W. M. (2002). Generalized linear discriminant sequence kernels for speaker recognition. In *Proceedings of Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference*, volume 1, pages I–161 –I–164.

[Dehak et al., 2010] Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., and Ouellet, P. (2010). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech and Language Processing*, pages 1 –1.

[D'Haro et al., 2012] D'Haro, L. F., Glembek, O., Plchot, O., Pavel Matějka, M. S., Cordoba, R., and Černocký, J. (2012). Phonotactic language recognition using i-vectors and phoneme posteriogram counts. In *Proceedings of Interspeech 2012*, volume 2012. To appear.

[Furui, 1986] Furui, S. (1986). Speaker-independent isolated word recognition using dynamic features of speech spectrum. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 34:52–59.

[Gales, 1999] Gales, M. (1999). Semi-tied covariance matrices for hidden Markov models. *IEEE Trans. Speech and Audio Processing*, 7:272–281.

[Garcia-Romero, 2011] Garcia-Romero, D. (2011). Analysis of i-vector length normalization in Gaussian-PLDA speaker recognition systems. In *Proc. of the International Conference on Spoken Language Processing (ICSLP)*.

[Glembek et al., 2011a] Glembek, O., Burget, L., Bümmer, N., Plchot, O., and Matějka, P. (2011a). Discriminatively trained i-vector extractor for speaker verification. In *Proceedings of Interspeech 2011*, volume 2011, pages 137–140.

[Glembek et al., 2009] Glembek, O., Burget, L., Dehak, N., Brümmer, N., and Kenny, P. (2009). Comparison of scoring methods used in speaker recognition with joint factor analysis. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 4057 –4060.

[Glembek et al., 2011b] Glembek, O., Matějka, P., and Burget, L. (2011b). Simplification and optimization of i-vector extraction. In *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, CZ.

[Glembek et al., 2008] Glembek, O., Matějka, P., Burget, L., and Mikolov, T. (2008). Advances in phonotactic language recognition. In *Proc. Interspeech 2008*, number 9, page 4.

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

[Kenny, 2010] Kenny, P. (2010). Bayesian speaker verification with heavy–tailed priors. In *Proc. of Odyssey 2010*, Brno, Czech Republic. http://www.crim.ca/perso/patrick.kenny, keynote presentation.

[Kenny et al., 2005] Kenny, P., Boulianne, G., Ouellet, P., and Dumouchel, P. (2005). Factor analysis simplified. In *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 637– 640, Toulouse, France.

[Kenny et al., 2007] Kenny, P., Boulianne, G., Oullet, P., and Dumouchel, P. (2007). Joint factor analysis versus eigenchannels in speaker recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 15(7):2072–2084.

[Kenny and Dumouchel, 2004] Kenny, P. and Dumouchel, P. (2004). Experiments in speaker verification using factor analysis likelihood ratios. In *Proceedings of Odyssey 2004*.

[Kenny et al., 2008] Kenny, P., Ouellet, P., Dehak, N., Gupta, V., and Dumouchel, P. (2008). A study of inter-speaker variability in speaker verification. *IEEE Transactions on Audio, Speech and Language Processing*, 16(5):980–988.

[Kockmann, 2012] Kockmann, M. (2012). *SUBSPACE MODELING OF PROSODIC FEATURES FOR SPEAKER VERIFICATION*. PhD thesis, Brno University of Technology.

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

[Lin et al., 2008] Lin, C.-J., Weng, R. C., and Keerthi, S. S. (2008). Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*.

[Marcel et al., 2010] Marcel, S., Cool, C. M., Matejka, P., Ahonen, T., Cernocky, J., Chakraborty, S., Balasubramanian, V., Panchanathan, S., Chan, C., Kittler, J., Poh, N., Fauve, B., Glembek, O., Plchot, O., Jancik, Z., Larcher, A., Lévy, C., Matrouf, D., Bonastre, J.-F., Lee, P. H., Hung, J. Y., Hung, Y. P., Wu, S. W., Machlica, L., Mason, J. S. D., Mau, S., Sanderson, C., Monzo, D., Albiol, A., Nguyen, H. V., Bai, L., Wang, Y., Niskanen, M., Turtinen, M., Nolazco-Flores, J. A., Garcia-Perera, L. P., Aceves-Lopez, R., Villegas, M., and Paredes, R. (2010). On the results of the first mobile biometry (MOBIO) face and speaker verification evaluation. In *Proceedings of the ICPR 2010 Contests*, Istanbul, Turkey.

[Matějka et al., 2006] Matějka, P., Burget, L., Schwarz, P., and Černocký, J. (2006). Brno University of Technology system for NIST 2005 language recognition evaluation. In *Proceedings of Odyssey 2006: The Speaker and Language Recognition Workshop*, pages 57–64.

[NIST, nda] NIST (n.d.a). The NIST speaker recognition evaluation. `http://www.itl.nist.gov/iad/mig/tests/spk/`.

[NIST, ndb] NIST (n.d.b). The NIST speaker recognition evaluation. `http://www.nist.gov/speech/tests/spk/index.htm`.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, 2nd edition.

[Pearlmutter, 1994] Pearlmutter, B. A. (1994). Fast exact multiplication by the Hessian. *Neural Computation*, 6:147–160.

[Pelecanos and Sridharan, 2006] Pelecanos, J. and Sridharan, S. (2006). Feature warping for robust speaker verification. In *Proceedings of Odyssey 2006: The Speaker and Language Recognition Workshop*, pages 213–218.

[Povey et al., 2005] Povey, D., Kingsbury, B., Mangu, L., Saon, G., Soltau, H., and Zweig, G. (2005). fMPE: Discriminatively trained features for speech recognition. In *Proceedings of ICASSP 2005. IEEE International Conference*, volume 1, pages 961 – 964.

[Prince and Elder, 2007] Prince, S. J. D. and Elder, J. H. (2007). Probabilistic linear discriminant analysis for inferences about identity. In *11th International Conference on Computer Vision*.

[S. Marcel (IDIAP),] S. Marcel (IDIAP), P. Matějka (BUT), P. T. U. MOBIO project - deliverable 6.4. `http://www.mobioproject.org/project/deliverables`.

[Schwarz et al., 2006] Schwarz, P., Matějka, P., and Černocký, J. (2006). Hierarchical structures of neural networks for phoneme recognition. In *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 325–328, Toulouse, France.

[Shampine, 2007] Shampine, L. F. (2007). Accurate numerical derivatives in MATLAB. *ACM Trans. Math. Softw.*

[Soufifar et al., 2011] Soufifar, M., Kockmann, M., Burget, L., Plchot, O., Glembek, O., and Svendsen, T. (2011). ivector approach to phonotactic language recognition. In *Proceedings of Interspeech 2011*, volume 2011, pages 2913–2916.

[Vair et al., 2007] Vair, C., Colibro, D., Castaldo, F., Dalmasso, E., and Laface, P. (2007). Loquendo - Politecnico di Torino's 2006 NIST speaker recognition evaluation system. In *Proceedings of Interspeech 2007*, pages 1238–1241.

[Vapnik, 1995] Vapnik, V. N. (1995). *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA.

[Young et al., 2006] Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X. A., Moore, G., Odell, J., Ollason, D., Povey, D., and et al. (2006). The HTK book.