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

Acknowledgments

I would like to thank Honza Černocký, and Lukáš Burget for being excelent supervisors during my studies. I was honored and priviledged each time I worked with them. Honza's push for education enabled me to attend numerous prestigous conferences and participate in wonderful projects and internships. Not only I got the opportunity to learn interesting things, but also to meet very interesting people and work with them. Endless discussions with Lukáš—interleaved with relaxing guitar jam-sessions—gave me important ideas and help for this thesis.

At the same time I have to thank all members of the Speech@FIT research group for their support and help, especially the Language and Speaker recognition team: Pavel Matějka, Olda Plchot, Petr Schwarz, Martin Karafiát, Karel Veselý, Sandro Cumani, Mehdi Soufifar, and our technical guru Tomáš Kašpárek. Not to forget, my credits also go to SPON—the unofficial "Fellowship of the Beloved Beverage"—for sharing the enjoyment of drinking coffee.

I'd also like to thank the participants and organizers of the JHU Speaker Workshop and the BOSARIS Workshop, from which most of the ideas for this thesis come from. Special thanks to Niko Brümmer and Patrick Kenny for the essential ideas and help during this work.

Thanks to all co-authors of the underlying papers for their support and kind agreement to share the text and ideas of the papers in this thesis.

Special thanks to Marcel Kockmann for sharing his ideas and for providing useful information for this work, especially introductory images and database description.

Very special thanks to my wife Maruška for her support and patience, thanks to my kids Margaréta and Augustýn for being with me, thanks to my parents and all the family for their moral support and helping out with my parental duties during the compilation of this work.

Contents

Nomenclature

ANN	Artificial Neural Network
BMBA	Bi-Modal Biometric Authentication
CDF	Cumulative Density Function
CG	Conjugate Gradient
DCF	Detection Cost Function
DCT	Discrete Cosine Transform
DET	Detection Error Tradeoff
DFT	Discrete Fourier Transform
EER	Equal Error Rate
EM	Expectation Maximization
FA	Fals Alarm
GMM	Gaussian Mixture Model
HLDA	Heteroscedastic Linear Discriminant Analysis
HMM	Hidden Markov Model
HTPLDA	Heavy-Tailed Probabilistic Linear Discriminant Analysis
JFA	Joint Factor Analysis
LDA	Linear Discriminant Analysis
LLR	Log-Likelihood Ratio
LVCSR	Large Vocabulary Continuous Speech Recognition
MCE	Minimum Classification Error
MFCC	Mel-Filterbank Cepstral Coefficients
ML	Maximum Likelihood

MMI	Maximum Mutual Information
MPE	Minimum Phoneme Error
NIST	National Institute of Standards and Technology
NN	Neural Network
PDF	Probability Density Function
PLDA	Probabilistic Linear Discriminant Analysis
ROC	Receiver Operating Characteristics
SRE	Speaker Recognition
STG	Short-Time Gaussianization
SVD	Singular Value Decomposition
SVM	Support Vector Machine
UBM	Universal Background Model
VAD	Voice Activity Detection
GSM	Global System for Mobile Communications
WCCN	Within-Class Covariance Normalization

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 pass-phrase, 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. Let us now take a look at the levels of information in which moder SRE systems operate:

1.1 Extracting Speaker Information

At the very beginning of an automatic speaker recognition, a speech waveform is provided to the system. According to [Reynolds, 2002], several levels of information can be found in speech (going from the lowest level to the highest):

- **acoustic**: spectral features conveying vocal tract information
- **prosodic**: features derived from prosody (pitch, energy, tracks, etc.) to characterize speaker-specific patterns [Kockmann, 2012]
- **phonetic**: analysis of sequences of phonemes specific to the speaker (see e.g. [Navrátil et al., 2003])
- **idiolect**: analysis of sequences of words
- **linguistic**: analysis of linguistic patters which characterize the speaker's conversation styles

The last two levels are rare in automatic SRE and are mostly exploited by experts in the forensic area.

In this work, I deal with the acoustic level of information, however many techniques described in this work were successfully used in other fields such as prosodic SRE [Kockmann, 2012].

1.1.1 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, nd]).

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.

1.1.2 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

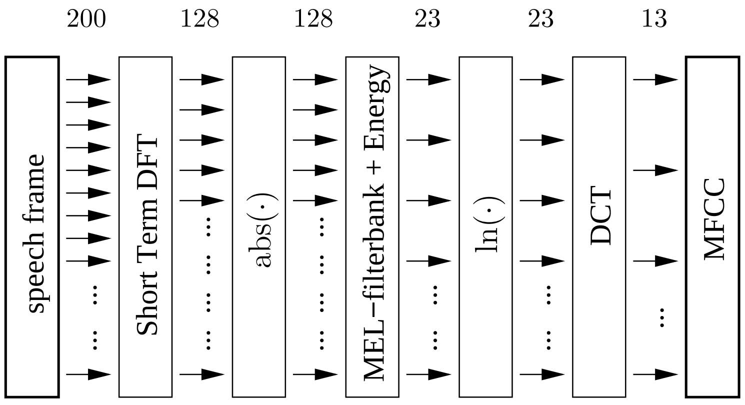


Figure 1.1: MFCC extraction—the numbers above show typical dimensionalities for frame length of 25ms at f_s of 8000Hz

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. Due to time progression, different experiments use slightly different steps and the details will be given in the corresponding chapters. However, let us now give a brief overview of the used methods.

Mel-Filterbank Cepstral Coefficients

Mel-Filterbank Cepstral Coefficients (MFCC's) have been standardly used in recent state-of-the-art SRE systems. The usage of the MFCCs in SRE has been inspired by the Large Vocabulary Continuous Speech Recognition (LVCSR) [Rabiner and Juang, 1993, Davis and Mermelstein, 1980]. They provide a short-term representation of the power spectrum of the analyzed signal. Figure 1.1 shows the extraction of the MFCC vector for one frame of speech and Figure 1.2 shows an example of the different stages for one frame. At first, the absolute value of the short-term Discrete Fourier Transform (DFT) is used to extract the amplitude of the spectrum. To emulate the human hearing aparatus, the spectrum of the signal is divided into frequency bands using the *Mel-Filterbank* [Rabiner and Juang, 1993] and energy for each band is computed as a sum of squared values of the amplitude spectrum (see Figure 1.2(c) for the visualization of the triangular-shaped, logarithmically spaced bank of filters). Note that the overall frame energy is computed as an average of squared samples. The logarithm of the band energies is computed to compensate for the dynamic range of the values and to emulate another non-linearity in the human hearing. Discrete Cosine Transform (DCT) is then used to de-correlate and reduce the dimensionality of the vector to get the final set of MFCC

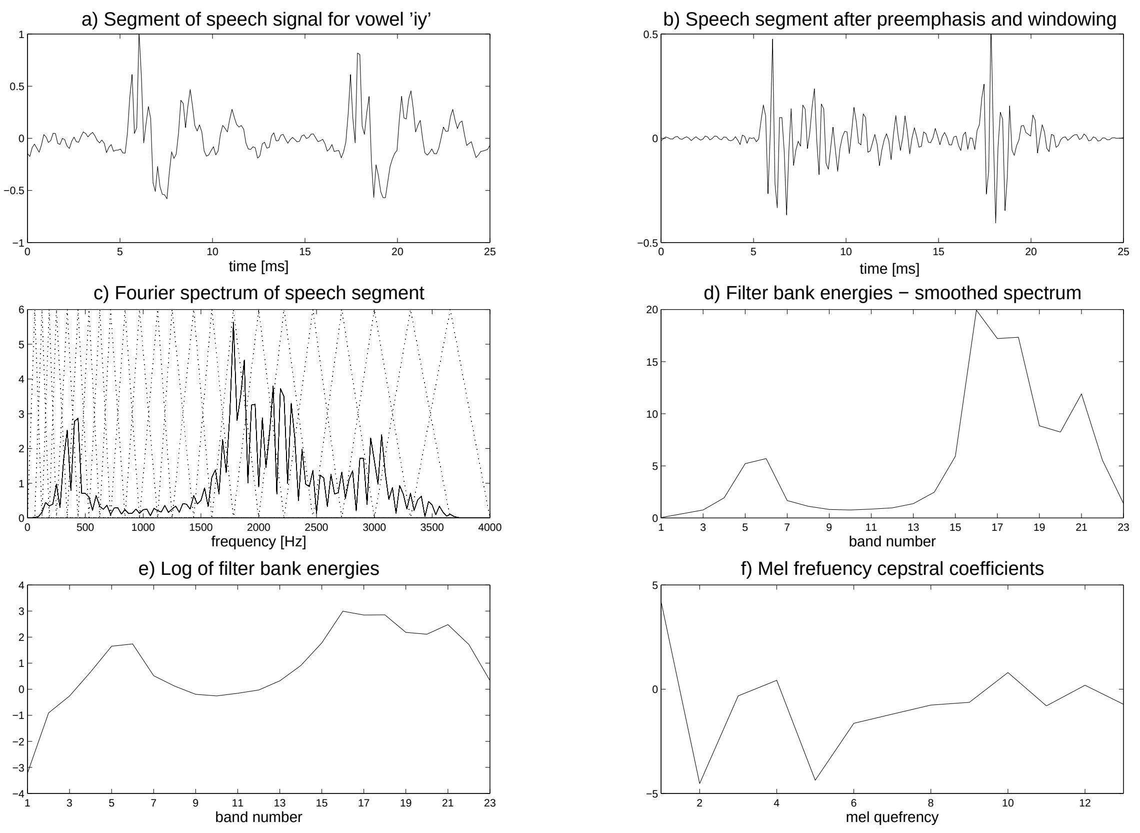


Figure 1.2: MFCC extraction—visualization of MFCC extraction steps. Reproduced from [Burget, 2004] with permission.

coefficients. Note that the zero-th coefficient contains information about the total frame energy and is therefore often discarded or replaced by the total energy itself.

Feature Derivatives

To capture the time progression, the consecutive feature vectors are usually extended with their 1st, 2nd, and/or 3rd order derivative approximations (higher orders are rarely used), commonly referred to as deltas, double-deltas, and triple-deltas [Furui, 1986]. The first order derivative for a feature vector $\mathbf{c}$ in frame k is usually realized as a linear combination of the $\pm N$ surrounding feature vectors, i.e.:

$$\Delta\mathbf{c}(k) = \sum_{j=-N}^{N} j\mathbf{c}(k-j), \tag{1.1}$$

where N is in our case set to 2. Higher-order derivatives can be obtained by recursively applying the above formula to the lower-order derivatives.

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

$$\hat{c}_{d,i}(k) = \frac{c_{d,i}(k) - \mu_{d,i}}{\sigma_{d,i}}, \tag{1.2}$$

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.

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.

1.2 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 1.3. 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

$$s_{\text{sym}} = \log \frac{p(x|\mathcal{H}_1)}{p(x|\mathcal{H}_2)} \tag{1.3}$$

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

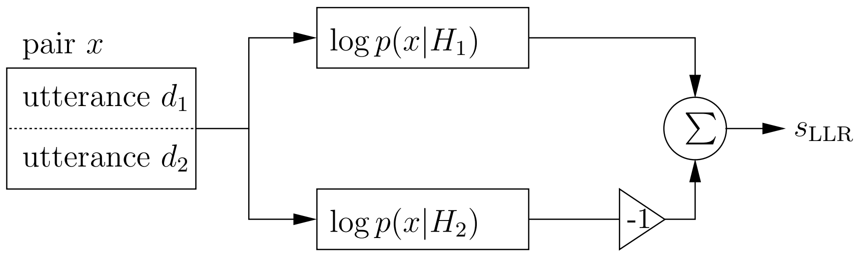


Figure 1.3: General symmetrical speaker verification procedure: an input trial x is given as a pair of utterances $\langle d_1, d_2 \rangle$ and the computation of the likelihood is conditioned by the hypotheses that the utterances come either from a single speaker ($\mathcal{H}_1$) or two different speakers ($\mathcal{H}_2$).

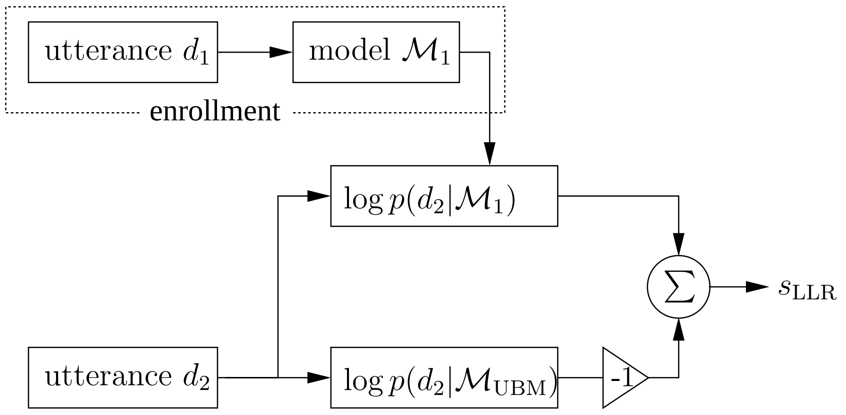


Figure 1.4: Asymmetrical speaker verification procedure: an input trial x is given as a model $\mathcal{M}_1$ and test utterance d_2 and the computation of the likelihood d_2 is conditioned by the model, i.e. either the probed speaker model $\mathcal{M}_1$ or the "the model of all speakers"—the UBM.

(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 $\mathcal{M}_{\text{UBM}}$, 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 (see Section 2.1). A diagram of such evaluation is shown in Figure 1.4. Mathematically, the asymmetrical problem is stated as

$$s_{\text{assym}} = \log \frac{p(d_2|\mathcal{M}_2)}{p(d_2|\mathcal{M}_{\text{UBM}})} \tag{1.4}$$

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

1.2.1 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., 2006b]. SVMs were used in place of $\mathcal{M}_1$ of Figure 1.4 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 6—discriminative training will address SRE as a symmetrical problem (as shown in Figure 1.3). 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.

1.2.2 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

$$s_{\text{norm}} = \frac{s - \mu}{\sigma}. \tag{1.5}$$

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

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

Figure 1.5 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 (1.5). Mathematically, we assure that

$$p\left(\frac{s_{\text{imp}} - \mu_d}{\sigma_d}\middle|d\right) = \mathcal{N}\left(\frac{s_{\text{imp}} - \mu_d}{\sigma_d}; 0, 1\right) \tag{1.7}$$

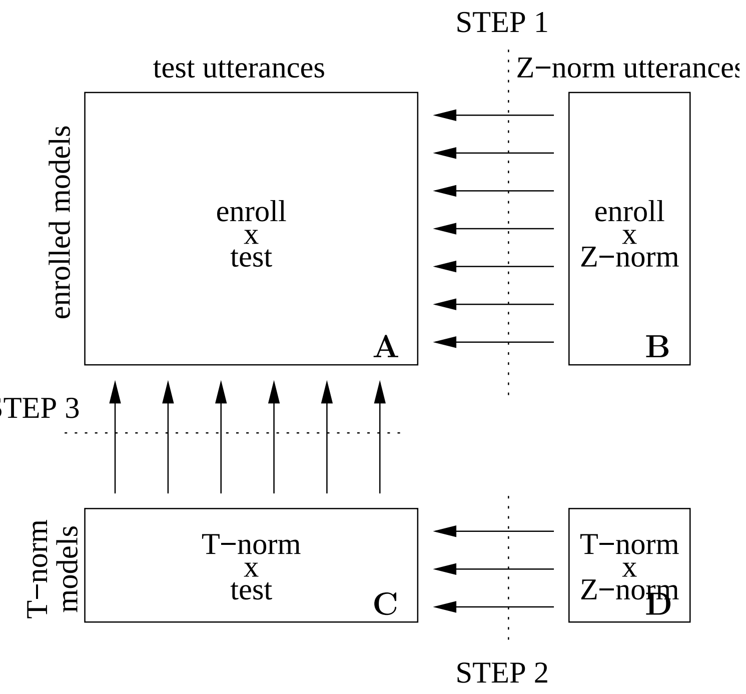


Figure 1.5: Application of ZT-norm. The boxes denote matrices of complete scores, i.e. all models against all scored utterances.

The method is marked as “STEP 3” in Figure 1.5. 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 1.5. 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 1.5.

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.

1.3 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 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].

1.3.1 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.

1.3.2 Structure of the thesis

The thesis is organized as follows:

- **Chapter 2** introduces the evaluation metrics, the data that are used throughout this work, and processing of the scores. The data is described in three contexts: databases, datasets, and feature extraction.
- **Chapter 3** outlines the basics of acoustic modeling and provides a theoretical introduction for the experiments of this work.

- **Chapter 4** presents the concept of Joint Factor Analysis and describes the work on different scoring methods, including the experimental results.
- **Chapter 5** presents the concept of i-vectors and introduces the simplification of i-vector extraction and i-vector extractor training.
- **Chapter 6** studies the technique of discriminative training of different parts of the speaker-recognition system and summarizes previous work. The new approach for i-vector extractor discriminative optimization together with the results are presented.

Chapter 2

Evaluation Metrics, Databases, and Datasets

Let us first get familiar with the evaluation metrics and the data that were used throughout the work. It is desirable to introduce and explain this issue at this point so that the reader gets familiarized with the terminology and qualities of the data. These will be referred to in the latter chapters.

2.1 Data sets

The procedure of building an SRE system consists of the following steps: first, the system is trained, then its parameters are eventualy adjusted, and finally the system is evaluated. All three steps require separate data sets, which have to be carefuly chosen, so that the system generalizes for unseed data. Respecting the order of the steps, the data sets are described as

- **training set** (also referred to as background set) — a large corpus for robust parameter estimation. This set usually contains several thousands of hours of speech and speakers and is used to train e.g. the UBM or hyper-parameters of the later described models such as JFA or PLDA.

- **development set** (also referred to as heldout set) — the system parameters are usually tuned using this set.

- **evaluation set** (also referred to as test set) — the final system performance is reported using this set. If the developped system generalizes well, the performance will positively corelate with the one measured on the development set.

2.2 Tests

2.2.1 NIST

Most experimental results are reported on the official NIST Speaker Recognition Evaluation tasks[1]. Because of time progression of this work, the experiments are reported on NIST SRE sets from years 2006 [NIST, 2006], 2008 [NIST, 2008], and 2010 [NIST, 2010].

Each of these evaluations consists of numerous tests, commonly referred to as *conditions*. These are defined by the properties of the audio data, such as channel, nominal length, number of utterances per trial side, etc. NIST usually defines some conditions as *core*, meaning they are mandatory for each participant of the evaluations. The results of this work are reported on the core conditions of the mentioned tests, as well as on some non-core conditions where needed. The individual databases are described in Section 2.4.

2.2.2 MOBIO Project and Target Platform

One of the parts of this work was to test the developped techniques on a mobile platform as a part of the MOBIO project. MOBIO was a consortium of universities, research centers and companies joined in the EU-funded project MOBIO[2]. The goal of this project was to bring robust biometric identification to the mobile devices with the common equipment, i.e. a microphone and a camera. The target research fields are therefore face recognition and speaker verification.

The project had two main scenarios, where the developed identification systems can be used:

- Embedded biometry where the Bi-Modal Biometric Authentication (BMBA) system is running entirely on a mobile phone. The system is designed to maximize the authentication performance and to minimize resources such as CPU, memory and speed.
- Remote biometry if the BMBA system needs too many resources to reach the required performance it will be hosted on a server while a minimum of essential functionalities would stay on the mobile phone such as capture, segmentation, preprocessing and feature extraction.

2.3 Evaluation Metrics

Let us now focus on how to evaluate the performance of speaker verification systems. As was mentioned earlier in Section 1.2, a speaker verification trial x is defined as a pair of two utterances $x = \langle d_1, d_2 \rangle$. A *supervised trial* is also associated with a label $h_x \in \{\mathcal{H}_1, \mathcal{H}_2\}$, depending on whether the two utterances come from the same speaker or two different speakers. A *test set*, usually denoted as $\mathcal{X}$, consists of a set of *same-speaker*

[1]National Institute of Standards and Technology (NIST), http://www.nist.gov/speech/tests/spk/index.htm

[2]`http://www.mobioproject.org`. The partners of the project were: Idiap Research Institute (CH), University of Manchester (UK), University of Surrey (UK), University of Avignon (FR), Brno University of Technology (CZ), University of Oulu (FN), IdeArk (CH), and Visidon (FI).

and *different-speaker* supervised trials $\mathcal{X}_1$ and $\mathcal{X}_2$, sometimes referred to as target and non-target trials, respectively. The task of a speaker verification system is to assign a correct label to the tested trial, i.e. classify the trial as $\mathcal{H}_1$ or $\mathcal{H}_2$. Two types of detection errors can arrise—*false alarms* (FA)—the recognizer classifies the different-speaker trial as same-speaker—and *missed detections* (Miss)[3]—the recognizer classifies the same-speaker trial as different-speaker. For a given test set, one can estimate the probabilities for the detection errors:

$$\begin{aligned} p(\text{miss}|\mathcal{X}) &= \frac{N_{\text{miss}}}{|\mathcal{X}_1|} \\ p(\text{fa}|\mathcal{X}) &= \frac{N_{\text{fa}}}{|\mathcal{X}_2|}, \end{aligned} \tag{2.1}$$

where $|\mathcal{X}_1|$ and $|\mathcal{X}_2|$ are the number of same- and different-speaker trials, respectively, and N_{fa} and N_{miss} are the number of false alarms and missed detections made by the system, respectively.

The raw output of the recognizer is a score[4], having higher value for same-speaker hypothesis and lower value for different-speaker hypothesis. The score is converted to hard decision by *thresholding.* Moving the threshold t balances between the two types of errors, letting the user choose the *operating point* of the system, i.e.

$$\begin{aligned} p(\text{miss}|\mathcal{X}) &= p(\text{miss}|\mathcal{X}, t) \\ p(\text{fa}|\mathcal{X}) &= p(\text{fa}|\mathcal{X}, t). \end{aligned} \tag{2.2}$$

2.3.1 DET Plot

In order to evaluate the system on a given dataset, it is desirable to visualize the errors for different thresholds. In the SRE community, The Detection Error Tradeoff (DET) graph is commonly used [Martin et al., 1997]. It is an alternative to a commonly used Receiver Operating Characteristics (ROC), and it plots the two types of errors on a non-linearily transformed x- and y- axes. An example of such plot is in Figure 2.1.

2.3.2 Equal Error Rate

It is often required to report a system performance using a single number instead of a complex figure such as the DET plot. In the SRE community, it is common to report the Equal Error Rate (EER)—an operating point where the two errors are equal. On a DET plot, the point is given as an intersection of the curve and the $x = y$ line. In Figure 2.1, the point is shown as hexagram mark. This is a very intuitive choice of an operating point, although not very useful in practical applications which usually require either very few miss detections (e.g., authorization systems), or false alarms (e.g., speaker spotting).

[3]The "FA" and "Miss" are addopted NIST conventions.

[4]preferabely log-likelihood ratio of the two hypotheses

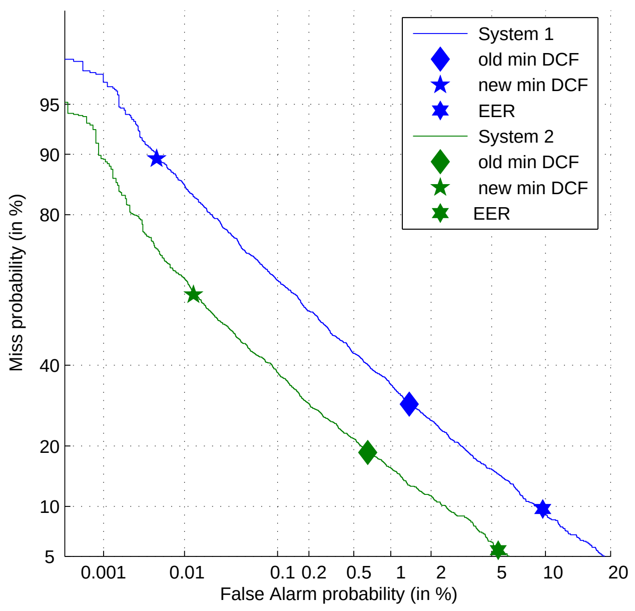


Figure 2.1: DET curves for two different systems. The three markers in each line correspond to the new DCF, the old DCF, and the EER, from left to right.

2.3.3 Detection Cost Function

NIST has introduced the Detection Cost Function (DCF) as a metric, which focuses on a particular operation point of interest. It is NIST's primary metric in its Speaker Recognition Evaluation series. It directly considers the overall costs based on the two types of errors. It is defined as a weighted sum of the false-alarm probability and the miss-detection probability:

$$\begin{aligned}\hat{C}_{\text{Det}} = C_{\text{miss}} \times p(\text{miss}|\mathcal{X}) \times p(\mathcal{H}_1) \\ + C_{\text{fa}} \times p(\text{fa}|\mathcal{X}) \times p(\mathcal{H}_2)\end{aligned} \tag{2.3}$$

with

$$p(\mathcal{H}_2) = 1 - p(\mathcal{H}_1) \ , \tag{2.4}$$

where C_{miss} and C_{fa} are the relative costs of the detection errors, and $p(\mathcal{H}_1)$ and $p(\mathcal{H}_2)$ are the prior probabilities for the trial being same- and different-speaker, respectively. The tripplet $\langle C_{\text{miss}}, C_{\text{fa}}, p(\mathcal{H}_1)\rangle$ defines the *application* for which the system is evaluated. The common values, as defined by NIST, are given in Table 2.1. Until the 2008, NIST had used values referred to as "old DCF" in the table. Since 2010, NIST has introduced new set of values, referred to as "new DCF", to emphasize the importance of very low false alarms. The points are shown in Figure 2.1 as diamonds and pentagrams, respectively.

To make the measure more intuitive, $\hat{C}_{\text{Det}}$ is further normalized by the best *a-priori* cost C_{Default}, i.e. one that would be obtained by setting all trials to either same- or

Table 2.1: *Common NIST DCF parameters (applications)*

	C_{fa}	C_{miss}	$p(\mathcal{H}_1)$
DCF_{old}	10	1	0.01
DCF_{new}	1	1	0.001

different-speaker, whichever is smaller:

$$C_{\text{Default}} = \min \begin{cases} C_{\text{miss}} \times p(\mathcal{H}_1) \\ C_{\text{fa}} \times p(\mathcal{H}_2) \end{cases} \tag{2.5}$$

and

$$\hat{C}_{\text{Norm}} = \hat{C}_{\text{Det}} / C_{\text{Default}}. \tag{2.6}$$

The cost is computed from the actual hard decisions, no matter how the threshold was chosen. An intuitive criterion for choosing the threshold is one that minimizes the DCF. One can compute a minimum possible DCF, referred to as *min-DCF*, to see the optimal threshold for the given test set:

$$\begin{aligned} \hat{C}^{\text{min}}_{\text{Det}} = \min_t \, & C_{\text{miss}} \times p(\text{miss}|\mathcal{X}, t) \times p(\mathcal{H}_1) \\ & + C_{\text{fa}} \times p(\text{fa}|\mathcal{X}, t) \times p(\mathcal{H}_2) \; . \end{aligned}$$

A more realistic measure, however, is the actual DCF or *act-DCF*, for which the threshold is chosen on some development set. The difference between the min-DCF and act-DCF reflects how well the system is *calibrated*, which will be further discussed in Section 2.3.6.

2.3.4 Constellation Plots

Alghough DET plot can show multiple curves for different systems, it can get confusing, when the curves are close to each other or if there are many systems to compare. Another usefull visualization tool is the *constellation plot*, where the systems are represented by pairs of chosen operating points and plotted in a 2D graph. An example of such plot is in Figure 2.2.

2.3.5 Application-Independent Metric

Assuming that the trial score is interpreted as a log-likelihood ratio—which is true for our systems by nature, as described in Section 1—an application-independent metric, referred to as $\hat{C}_{\text{llr}}$ has been proposed in [Brümmer and du Preez, 2006]. For a trial set $\mathcal{X}$ being composed of same-speaker trials $\mathcal{X}_1$ and different-speaker trials $\mathcal{X}_2$, the function is defined as

$$\hat{C}_{\text{llr}}(\mathcal{X}) = \frac{1}{2\log 2} \left(\frac{1}{|\mathcal{X}_1|} \sum_{x \in \mathcal{X}_1} \log(1 + e^{s_x^{-1}}) + \frac{1}{|\mathcal{X}_2|} \sum_{x \in \mathcal{X}_2} \log(1 + e^{s_x}) \right) \tag{2.7}$$

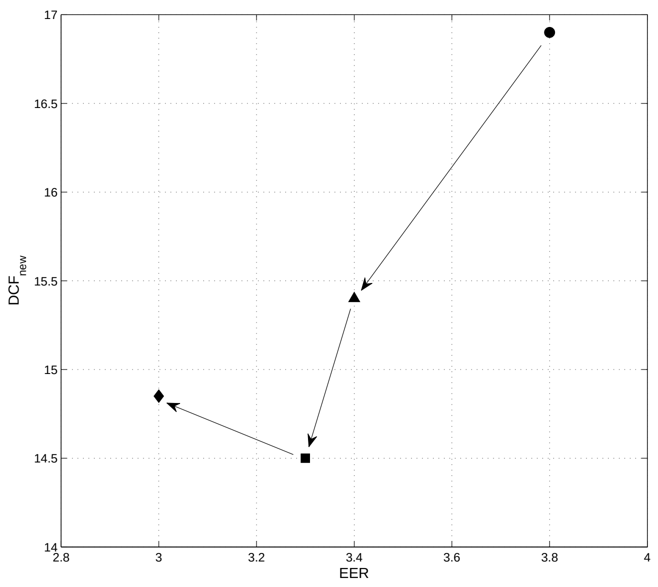


Figure 2.2: Example of a constellation plot. The plot shows the configuration of four systems for new DCF and EER. The arrows can be used to show the evolution of the system development on the important operating points.

where s_x is the score for trial x, and $|\mathcal{X}_1|$ and $|\mathcal{X}_2|$ represent the number of same- and different-speaker trials, respectively. Assuming the scores are log-likelihood ratios, and the prior probabilities for both trial classes are equal, the logarithms in the sums represent log-posterior probabilities of the trial being recognized as same-speaker of different-speaker, respectively, i.e.

$$\begin{aligned} s_x &= \operatorname{logit}(p_x) = \log \frac{p_x}{1-p_x} \\ p_x &= \operatorname{logit}^{-1}(s_x) = \frac{1}{1-e^{-s_x}} \end{aligned} \tag{2.8}$$

where, for convenience, we use $p_x = p(\mathcal{H}_1|x) = 1 - p(\mathcal{H}_2|x)$.

Note that we can interpret this metric as a *logistic-regression* objective function, i.e., a sum of logarithmic posterior probabilities of all trials being recognized correctly. This will be described in detail in Section 6.1.

2.3.6 Calibration

As mentioned earlier in Section 2.3, threshold has to be chosen in order to make the decision for a specified operating point. This can be achieved by *calibrating* the output scores of the detector to represent proper class (hypothesis) posterior probabilities, using the desired target prior in the calibration training. Calibration is understood as an affine transformation of the score. Each calibrated (logarithmic) score greater than the log of the target prior exceeds the threshold and is accepted in the requested operating point.

The process of calibration is described in [Brümmer and du Preez, 2006]. It is realized using logistic-regression which aims at maximizing the posterior probabilities of binary classification, which is in fact defined by (2.7).

It is also worth mentioning that stacking scores from multiple systems into a single vector and applying logistic regression to such data leads to discriminatively-trained score-level fusion.

2.4 Databases

This section contains the description of the data corpora that were used to build the datasets as described in Section 2.1.[5]

2.4.1 Switchboard

Switchboard 2 Phase II [Graff et al., 1999] was released in 1999 and consists of 4,472 five-minute telephone conversations involving 679 participants which were mainly recruited from US college campuses. Each speaker participated in at least 10 calls. *Switchboard 2 Phase III* [Graff et al., 2002] was recorded between 1997 and 1998 in the American South and consists of 2,728 calls from 640 participants (292 Male, 348 Female) which are all native English speakers. Both of these corpora only consist of landline calls. *Switchboard Cellular Part 1* [Graff et al., 2001] was recorded until 2000 and mainly focuses on cellular phone technology. It consists of 1,309 calls, or 2,618 sides (1,957 GSM), from 254 participants (129 Male, 125 Female), under varied environmental conditions. *Switchboard Cellular Part 2* [Graff et al., 2004] was released in 2004 and consists of 2,020 calls, or 4,040 sides (2,950 cellular, 2,405 female, 1,635 male), from 419 participants.

2.4.2 NIST SRE 2004

The NIST SRE 2004 corpus [Martin and Przybocki, 2004] consists of 10,743 telephone call segments recorded from 480 participants (181 Male, 299 Female) over landline as well as cellular phones.

2.4.3 NIST SRE 2005

The NIST SRE 2005 [NIST, 2005] corpus consists of 16,537 telephone call segments recorded from 528 participants (219 Male, 308 Female) over landline as well as cellular phones. Additionally, telephone calls were recorded over auxiliary microphones of eight different kinds. For both corpora, many segments have different lengths (from 10 seconds up to five minutes) but may stem from the same original full conversation. Furthermore, some segments contain summed conversations. Only unique full conversations with separate channel per speaker are used in the setup. Apart from native speakers, both collections also consist of non-native English and several foreign languages.

[5]The description of the databases was taken from [Kockmann, 2012] and [Pešán, 2011] with kind permission of the authors.

2.4.4 NIST SRE 2006

This corpus is used for many early experiments that are reported during the thesis. However, for experiments on NIST SRE 2008 and 2010, the corpus has also been included into the background data set.

Overall, the NIST SRE 2006 corpus [NIST, 2006] consists of 24,637 telephone call segments recorded from 1089 participants (462 Male, 626 Female) over landline as well as cellular phones. Additionally, telephone calls were recorded over auxiliary microphones of eight different kinds. Again, many segments have different lengths (from 10 seconds up to five minutes) but may stem from the same original full conversation. Furthermore, some segments contain summed conversations. Only unique full conversations with separate channel per speaker are used in the setup. Again, native as well as non-native English and several foreign languages are recorded. Special attention has to be paid while using this data, as recordings from the NIST SRE 2005 corpus have been recycled.

Experiments that report on the NIST 2006 corpus are always performed on the core condition which contains English trials only. The 1-side training 1-side test condition is considered, where approximately 2.5 minutes of speech (from a 5-minute telephone conversation) are available to train each speaker and for each test utterance. This set contains 329 female and 248 male training utterances (multiple utterances can be produced by one distinct speaker), 1,846 target trials, and 21,841 nontarget trials.

2.4.5 NIST SRE 2008

In the 2008 evaluation [NIST, 2008], NIST broadened the scope of the evaluation by introducing interview speech that was recorded over several microphones. As a consequence, even the core condition (only full five minute calls in English speech) contains different sub-conditions involving different types of speech or channels during both speaker enrollment and verification. In this thesis, the results on the 2008 corpus are reported for the following conditions: *tel-phn:tel-phn* uses only conversational telephone speech of full calls for enrollment and verification, with 1,154 target and 1,516,837 nontarget trials (equivalent to the preceding years). *int-mic:tel-phn* uses interview speech recorded over several microphone types for enrollment and conversational telephone calls for verification, with 1,459 target and 820,215 nontarget trials. The condition *int-mic:int-mic* uses interview speech recorded over microphone for both enrollment and verification, consisting of 33,743 target and 1,108,882 nontarget trials.

2.4.6 NIST SRE 2010

Finally, results are reported on selected conditions of the NIST 2010 extended evaluation [NIST, 2010], that match the conditions in the 2008 development set: *tel-phn:tel-phn* uses only conversational telephone speech of full calls for enrollment and verification with 7,169 target and 408,950 nontarget trials (official extended condition 5). *int-mic:tel-phn* uses interview speech recorded over several microphone types for enrollment and conversational telephone calls for verification with 3,989 target and 637,850 nontarget trials (official extended condition 3). The condition *int-mic:int-mic* uses interview speech recorded over microphones for both enrollment and verification, consisting of 15,084 target and 2,789,534 nontarget trials (official extended condition 2).

2.4.7 Fisher English

Fisher English is a collection of conversational telephone speech collected in 2003 by LDC. The database protocol was created at LDC to address a critical need of developers trying to build robust ASR systems. A very large number of participants each make a few calls of short duration speaking to other participants (in English), whom they typically do not know, about assigned topics. This maximizes inter-speaker variation and vocabulary breadth although it also increases formality. The database contains 11,699 recorded telephone conversations, each lasting up to 10 minutes.

2.4.8 MOBIO

This bi-modal database was captured in two phases and consists of 152 participants (100 males and 52 females). Each session recorded for Phase I consists of 21 questions which the user was prompted to answer. These questions varied from set responses, read speech from a paper through to, to free speech. It contains 1650 target and 34650 nontarget female trials. On male part, it contains 2925 target and 111150 nontarget trials.

During the second phase (Phase II), six more sessions were recorded from 152 participants[6]. Each session for Phase II consists of 11 questions and includes the same variation as the one captured in Phase I. [Hadid and McCool, 2010] It contains 196 target and 3332 nontarget female trials, and 130 target and 4810 nontarget male trials. MOBIO data were recorded in normal conditions with slight noise on the background. The average length of the utterances is approximately 5 seconds.

[6]Compared to the first phase, eight participants were not able to take part in the second phase of the recording

Chapter 3

Gaussian Mixture Modeling

"A mixture model is a probabilistic model for representing the presence of sub-populations within an overall population, without requiring that an observed data-set should identify the sub-population to which an individual observation belongs" [Wikipedia, nd]. In other words a mixture model of C *mixture components* assumes that each (multivariate) data point $\mathbf{o}_i$ has been generated by one mixture component c, formally

$$\mathbf{o}_i \sim p(\mathbf{o}|c). \tag{3.1}$$

where $p(\mathbf{o}|c)$ is the probability density function (PDF) for the given component. However, the identity of the component c is a hidden variable and as such, a prior probability $p(c)$—referred to as mixture *weight* $w^{(c)}$—is imposed on it. The mixture probability density function is then given as

$$p(\mathbf{o}) = \sum_{c=1}^{C} w^{(c)} p(\mathbf{o}|c). \tag{3.2}$$

Gaussian mixture models (GMM) are a family of mixture models where the PDF for each mixture is a Gaussian distribution. They have been the essential part of modern speaker recognition (for essentials, see [Douglas Reynolds, 2000]), as well as of other fields of speech processing, e.g. language identification (e.g. [Torres-Carrasquillo et al., 2002]), LVCSR (e.g. [Young et al., 2006]), etc.

The probability density function for a single F-dimensional multivariate Gaussian distribution (see e.g. [Bishop, 2006]) is given as:

$$\mathcal{N}(\mathbf{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{F/2} |\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\mathbf{o}-\boldsymbol{\mu})} \tag{3.3}$$

where $\boldsymbol{\mu}$ is the vector of the *mean* and $\boldsymbol{\Sigma}$ is the *covariance* matrix[1]. Substituting (3.4) into (3.2), the probability density function of a GMM for a data vector $\mathbf{o}$ is given as:

$$\mathcal{G}(\mathbf{o}; \theta) = \sum_{c=1}^{C} w^{(c)} \mathcal{N}\left(\mathbf{o}; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}\right), \tag{3.4}$$

[1]Note that the symbols $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ will have slightly different meaning later in the text; I have chosen to use this notation to keep consistent with most of the literature.

with parameters

$$\theta = \langle \boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{\Sigma} \rangle \tag{3.5}$$

where C is the number of Gaussian components, $\boldsymbol{w}$ is the vector of weights for the corresponding mixture components,

$$\boldsymbol{w} = \begin{bmatrix} w^{(1)} \\ \vdots \\ w^{(C)} \end{bmatrix}, \tag{3.6}$$

$\boldsymbol{\mu}$ is the *supervector*[2] of concatenated component-level mean vectors $\boldsymbol{\mu}^{(c)}$:

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}^{(1)} \\ \vdots \\ \boldsymbol{\mu}^{(C)} \end{bmatrix}, \tag{3.7}$$

and $\boldsymbol{\Sigma}$ is generally a block-matrix of per-component covariance matrices:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}^{(1)} & 0 & \cdots & 0 \\ 0 & \boldsymbol{\Sigma}^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{\Sigma}^{(C)} \end{bmatrix}. \tag{3.8}$$

In most cases, however, diagonal version of the Gaussian distribution is used due to simplicity and CPU power, so $\boldsymbol{\Sigma}$ will be a pure diagonal matrix with the corresponding variances mapped to the diagonal. An illustration of a GMM for 2-dimensional data is shown in Figure 3.1.

3.1 Data Alignment

As was mentioned earlier, the identity of the mixture component for each data point is hidden, but having observed the data point, posterior probabilities $p(c|\mathbf{o}_i)$—also referred to as *occupation probabilities* and shortly denoted as $\gamma_i^{(c)}$—can be computed using the Bayes rule:

$$\gamma_i^{(c)} = \frac{w^{(c)}\mathcal{N}\left(\mathbf{o}_i; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}\right)}{\sum_{c=1}^{C} w^{(c)}\mathcal{N}\left(\mathbf{o}_i; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}\right)}. \tag{3.9}$$

The configuration of the posterior probabilities for each frame is referred to as the *alignment* of the data to the mixture components.

[2]The term supervector is used to denote that the "large" vector was constructed by concatenation of smaller vectors. It will be used frequently throughout this work.

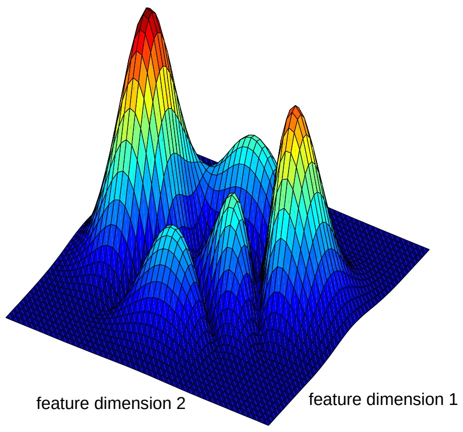


Figure 3.1: GMM probability density function for 2-dimensional data

3.2 Sufficient Statistics

Most of the formulae throughout the rest of the work will be expressed in terms of the *sufficient statistics*. They provide the complete information about the given utterance needed to compute any estimate of the parameters of a GMM [Bishop, 2006].

Let us assume, that an utterance d is given as a sequence of N feature vectors of dimensionality F, i.e. it is represented as a $F \times N$ matrix $\mathbf{O} = [\mathbf{o}_1 \dots \mathbf{o}_N]$. Having the alignment of each frame, as defined by (3.9), the statistics are defined as

$$N^{(c)} = \sum_{i=1}^{N} \gamma_i^{(c)} \tag{3.10}$$

$$\mathbf{f}^{(c)} = \sum_{i=1}^{N} \gamma_i^{(c)} \mathbf{o}_i \tag{3.11}$$

$$\mathbf{S}^{(c)} = \sum_{i=1}^{N} \gamma_i^{(c)} \mathbf{o}_i \mathbf{o}_i{}' \; . \tag{3.12}$$

We refer to these as the zero-, the first-, and the second-order statistics (or cumulants) respectively. For the sake of convenience, let us rewrite the statistics in the expanded form of supervector and supermatrix as:

$$\mathbf{N} = \begin{bmatrix} N^{(1)}\mathbf{I} & 0 & \cdots & 0 \\ 0 & N^{(2)}\mathbf{I} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & N^{(C)}\mathbf{I} \end{bmatrix}$$

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}^{(1)} \\ \vdots \\ \mathbf{f}^{(C)} \end{bmatrix} \tag{3.13}$$

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}^{(1)} & 0 & \cdots & 0 \\ 0 & \mathbf{S}^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{S}^{(C)} \end{bmatrix},$$

where the identity matrices in (3.13) have the same dimensionality as the feature vector.

3.3 The Likelihood Function

Let us now take a look at the different forms of the GMM likelihood function since they will be used later in this work. Assuming that the consecutive frames are statistically independent, the likelihood of the data, given the model parameters θ, is given as

$$p(\mathbf{O}|\theta) = \prod_{i=1}^{N} \mathcal{G}(\mathbf{o}_i;\theta) \; . \tag{3.14}$$

Usually, the logarithm of the likelihood is required, e.g. for scoring as described in Section 1.2, or when deriving the function to estimate the parameters. Its basic form is given as

$$\log p(\mathbf{O}|\theta) = \sum_{i=1}^{N} \log \sum_{c=1}^{C} w^{(c)} \mathcal{N}\left(\mathbf{o}_i; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}\right) . \tag{3.15}$$

We can now use slightly more complicated form of the log-likelihood (as derived by (A.12) in Appendix A.1) and rewrite it as

$$\begin{aligned} \log p(\mathbf{O}|\theta) = & \sum_{i=1}^{N}\sum_{c=1}^{C} q(c) \log \mathcal{N}\left(\mathbf{o}_i; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}\right) \\ & - \sum_{i=1}^{N}\sum_{c=1}^{C} q(c) \log \frac{q(c)}{w^{(c)}} \\ & + \sum_{i=1}^{N} \mathrm{D}_{\mathrm{KL}}\left(q(c) \| \gamma_i^{(c)}\right) \; , \end{aligned} \tag{3.16}$$

where $q(c)$ is an arbitrary distribution over the GMM components, i.e. arbitrary alignment. If we set it to the true posterior probabilities $q(c) = p(c|\mathbf{o}_i) = \gamma_i^{(c)}$, the KL divergence in (3.16) vanishes, and we can reformulate the likelihood function in terms of the posterior probabilities as

$$\log p(\mathbf{O}|\theta) = \sum_{i=1}^{N}\sum_{c=1}^{C} \gamma_i^{(c)} \log \mathcal{N}\left(\mathbf{o}_i; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}\right) - \sum_{i=1}^{N}\sum_{c=1}^{C} \gamma_i^{(c)} \log \frac{\gamma_i^{(c)}}{w^{(c)}} \,. \tag{3.17}$$

Comparing the two expressions, (3.15) and (3.17), note that the sum over mixture components has moved out of the logarithm. This generally allows us to represent the first term using the sufficient statistics:

$$\begin{aligned}\log p(\mathbf{O}|\theta) = &\sum_{c=1}^{C} N_c \log \frac{1}{(2\pi)^{F/2}|\boldsymbol{\Sigma}_c|^{1/2}} \\ &- \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}) \\ &+ \boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\mathbf{f} - \frac{1}{2}\boldsymbol{\mu}'\mathbf{N}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ &- \sum_{i=1}^{N}\sum_{c=1}^{C} \gamma_i^{(c)} \log \frac{\gamma_i^{(c)}}{w^{(c)}} \,.\end{aligned} \tag{3.18}$$

3.3.1 Fixed Alignment

Looking back to (3.16), $q(c)$ is an arbitrary distribution of the GMM components. Setting it to anything different than the true model alignment makes the KL-divergence term positive. Let us now assume that the true alignment is missing and that it is provided via different model parametrized by θ_γ. We can set $q(c) = p(c|\mathbf{o}_i, \theta_\gamma)$. This, however, makes the computation of the KL-divergence impossible, since it requires the true alignment. If we assume that the provided alignment is "close-enough" to the true alignment, we can omit the KL-divergence term and approximate the true log-likelihood function by

$$\begin{aligned}\log \tilde{p}(\mathbf{O}|\theta, \theta_\gamma) = &\sum_{c=1}^{C} N_c(\theta_\gamma) \log \frac{1}{(2\pi)^{F/2}|\boldsymbol{\Sigma}_c|^{1/2}} \\ &- \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}^{-1}\mathbf{S}(\theta_\gamma)\right) \\ &+ \boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\mathbf{f}(\theta_\gamma) - \frac{1}{2}\boldsymbol{\mu}'\mathbf{N}(\theta_\gamma)\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ &- \sum_{i=1}^{N}\sum_{c=1}^{C} \gamma_i^{(c)}(\theta_\gamma) \log \frac{\gamma_i^{(c)}(\theta_\gamma)}{w^{(c)}} \,,\end{aligned} \tag{3.19}$$

which has essentially the same form as (3.18), however, the alignment (and therefore the sufficient statistics) are parametrized by the model which was used for their computation. Note that the approximation is a lower-bound to the true log-likelihood function as the omitted KL-divergence is always non-negative.

A reasonably good alignment can be obtained using the Universal Background Model (UBM) whose purpose is to model "any" speaker, as described in Section 1.2. As will be shown in Section 4.2, this is a good approximation.

Let us recall, that the asymmetrical scoring of a trial (as defined by (1.4)) is given as a log of the ratio of the likelihood of the data given the speaker model $\mathcal{M}$ and the likelihood of the data given the UBM $\mathcal{M}_{\text{UBM}}$, which can be rewritten as

$$s_{\text{assym}} = \log p(\mathbf{O}|\mathcal{M}) - \log p(\mathbf{O}|\mathcal{M}_{\text{UBM}}) \ . \tag{3.20}$$

Fixing the alignment to the UBM gives an approximation of the score:

$$\begin{aligned} \tilde{s}_{\text{assym}} &= \log \tilde{p}(\mathbf{O}|\mathcal{M}, \mathcal{M}_{\text{UBM}}) - \log \tilde{p}(\mathbf{O}|\mathcal{M}_{\text{UBM}}, \mathcal{M}_{\text{UBM}}) & (3.21)\\ &= \log \tilde{p}(\mathbf{O}|\mathcal{M}, \mathcal{M}_{\text{UBM}}) - \log p(\mathbf{O}|\mathcal{M}_{\text{UBM}}) \ , & (3.22) \end{aligned}$$

where we notice that the UBM alignment makes the UBM produce true log-likelihoods and therefore, the approximated log-likelihood ratio is a lower-bound to the true log-likelihood ratio.

Note that the UBM-alignment assumption is generally used throughout this work. Most of the formulae are given in terms of the sufficient statistics which were computed using the UBM alignment.

3.4 Maximum Likelihood Estimation of the Parameters

Generally, the Maximum Likelihood (ML) estimate of the parameters θ is given as

$$\theta_{\text{ML}} = \arg\max_{\theta} p(\mathbf{O}|\theta) \ , \tag{3.23}$$

i.e., we maximize the likelihood of some training data $\mathbf{O}$. It is usually found by solving

$$\frac{\mathrm{d} \log p(\mathbf{O}|\theta)}{\mathrm{d}\,\theta} = 0, \tag{3.24}$$

however, no closed-form solution exists for GMM. The model is usually trained iteratively using the Expectation Maximization (EM) procedure [Dempster et al., 1977, Bishop, 2006]. A brief mathematical introduction to the method is in Appendix A.2. In the E-step, an auxiliary function $\mathcal{Q}_{\text{GMM}}$—as a lower-bound to the real log-likelihood function—is constructed by fixing the alignment of the data using the current model estimate θ_0 (the sufficient statistics are collected via the posteriors computed using θ_0). The solution is technically equivalent to (3.19):

$$\mathcal{Q}_{\text{GMM}}(\theta, \theta_0) = \tilde{p}(\mathbf{O}|\theta, \theta_0). \tag{3.25}$$

In the M-step of the EM algorithm, the new ML estimate of parameters is computed as

$$\theta_{\text{ML}} = \arg\max_{\theta} \mathcal{Q}_{\text{GMM}}(\theta, \theta_0) \tag{3.26}$$

for which closed-form solutions exist:

$$\begin{aligned}\boldsymbol{\mu}_{\mathrm{ML}}^{(c)} &= \frac{1}{N^{(c)}}\mathbf{f}^{(c)} \\ \boldsymbol{\Sigma}_{\mathrm{ML}}^{(c)} &= \frac{1}{N^{(c)}}\mathbf{S}^{(c)} - \boldsymbol{\mu}_{\mathrm{ML}}^{(c)}\boldsymbol{\mu}_{\mathrm{ML}}^{(c)\,\prime} \\ w_{\mathrm{ML}}^{(c)} &= \frac{N^{(c)}}{N}\ ,\end{aligned} \tag{3.27}$$

where the statistics are functions of θ_0. Repeating the E and M steps guarantees not to decrease the likelihood.

ML can be used when sufficient amount of data is available. Considering the asymmetrical SRE approach—as described in Section 1.2, Figure 1.4—the UBM is usually trained this way. By having a large training set, it is assured that the ML estimation of the parameters is robust enough (see e.g. [Burget et al., 2007]). As for estimating speaker models $\mathcal{M}_1$, there is usually not enough data to estimate all of the GMM parameters correctly without overfitting. The common practice is to compute the $\boldsymbol{\mu}$ parameter only (see e.g. [Douglas Reynolds, 2000]). Therefore, the speaker identity (and the speaker model $\mathcal{M}_1$) is given strictly by $\boldsymbol{\mu}$, while $\boldsymbol{w}$ and $\boldsymbol{\Sigma}$ are shared among all models and are taken from the (robustly estimated) UBM which allows us to omit the second term from (3.18).

3.4.1 UBM Training

The UBM is typically trained in the ML fashion since there is usually large amount of training data available. Note that the GMM has to be initialized in order to apply the EM algorithm. There are various approaches to initializing the parameters, such as using K-means clustering, random initialization, etc. In our work, we used progressive Gaussian splitting, i.e. starting by a single-Gaussian model (which has closed form solution), the components are duplicated and shifted by a small step in the direction of the largest variability after certain number of EM iterations. The training scheme that was used in this work is given in Algorithm 1. Note that covariance flooring is optionally performed [Young et al., 2006] to prevent overfitting of the estimation, i.e. if the distribution is found too sharp, the covariance is forced to stay within some trusted limits. In case of diagonal covariance matrix model, scaled average covariance matrix, taken over all GMM components, is often used as a threshold. In the case of full covariance models, the flooring can be performed, e.g., via Singular Value Decomposition (SVD) [Povey et al., 2011, Algorithm 5.3]. Figure 3.2a shows the result of an example of training GMM parameters when lots of data is available. If little data is available, as seen in Figure 3.2b, we see that the covariance matrices do not cover the data completely and the estimation suffers from overfitting.

3.5 MAP adaptation

Another approach to computing the GMM parameters is using the *maximum a-posteriori* criterion (MAP). Generally, the parameters are computed as

$$\theta_{\mathrm{MAP}} = \arg\max_{\theta} p(\theta|d), \tag{3.28}$$

Algorithm 1: General maximum likelihood GMM training algorithm

```
Data: Data from training set
Result: Model $\mathcal{M}$
begin
    $C \leftarrow 1$;
    compute $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ for single Gaussian model;
    while not enough mixtures do
        duplicate each Gaussian $c$ into $c_1$ and $c_2$ ;
        find the largest eigen-value and its corresponding eigen-vector $\mathbf{r}$ of $\boldsymbol{\Sigma}^{(c)}$ ;
        add $0.2\mathbf{r}$ to $\boldsymbol{\mu}^{(c_1)}$ ;
        add $-0.2\mathbf{r}$ to $\boldsymbol{\mu}^{(c_2)}$ ;
        for $i \leftarrow 1$ to number of iterations do
            do the E step;
            do the M step;
            optional: do covariance flooring;
```

where $p(\theta|d)$ is the posterior probability for the parameters θ given the input d:

$$p(\theta|d) = \frac{p(d|\theta)p(\theta)}{p(d)} \tag{3.29}$$

This way, we select the most likely parameters. Note that the denominator of the formula does not depend on the parameters and can therefore be omitted for optimization. The MAP criterion is then given as

$$\mathcal{Q}_{\text{MAP}}(\mathbf{O}|\theta) = p(\mathbf{O}|\theta)p(\theta) \ , \tag{3.30}$$

i.e. it is an ML estimation with a prior distribution imposed on the parameters. This approach is helpful if very little data is available and more importantly if we choose a good prior. In other words, ML is a special case of MAP, where flat priors are considered, which gives good parameter estimation for large training data sets. If we take look at the asymmetrical approach again (as defined in Section 1.2), where the speaker model $\mathcal{M}$ has to be trained from little data, MAP is a good candidate to train the model parameters robustly.

Let us now see, how the UBM can be used for "adapting"[3] $\boldsymbol{\mu}$. The issue with MAP is the choice of the prior $p(\boldsymbol{\mu})$. Again, we only work with $\boldsymbol{\mu}$ and the rest of the parameters are shared with the UBM, and fixed alignment to UBM is assumed. Since the resulting probability density function is going to be per-component Gaussian, the conjugate per-component priors to the likelihood are —thanks to fixed alignment—again Gaussians. Instead of estimating the parameters of the priors, we approximate them using the UBM as:

$$p\left(\boldsymbol{\mu}^{(c)}\right) = \mathcal{N}\left(\boldsymbol{\mu}^{(c)}; \boldsymbol{\mu}^{(c)}_{\text{UBM}}, \frac{1}{\tau^2}\boldsymbol{\Sigma}^{(c)}_{\text{UBM}}\right) , \tag{3.31}$$

[3] "UBM adaptation" is a widely used term which refers to using UBM to choose the prior for the MAP estimation.

where τ is an adaptation constant to be chosen by the user, and its interpretation is clarified in the following explanation. This approximation has been proposed e.g. in [Douglas Reynolds, 2000, Young et al., 2006]. Thanks to fixed alignment, estimation of $\boldsymbol{\mu}^{(c)}$ has a closed-form solution, given as

$$\boldsymbol{\mu}_{\text{MAP}}^{(c)} = \beta^{(c)} \boldsymbol{\mu}_{\text{ML}}^{(c)} + \left(1 - \beta^{(c)}\right) \boldsymbol{\mu}_{\text{UBM}}^{(c)} \tag{3.32}$$

with

$$\beta^{(c)} = \frac{N^{(c)}}{N^{(c)} + \tau}, \tag{3.33}$$

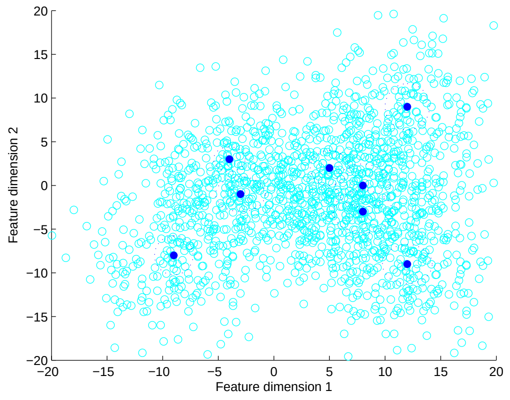
where $\boldsymbol{\mu}_{\text{ML}}^{(c)}$ is the ML estimate of the mean (given that the weights and covariances are fixed and set to the UBM values, and the Gaussian alignment is assumed to be the same as the UBM's), $\boldsymbol{\mu}_{\text{UBM}}^{(c)}$ is the mean of the UBM, and $N^{(c)}$ are zero-order statistics. We see that the τ constant controls the linear combination of the two models' parameters. Knowing that it is added to the data mass to control this weighing, one way to explain its meaning is e.g. by stating that it takes τ frames to move the parameter values half way between the UBM and the ML estimate. Figure 3.2c shows the behavior of MAP adaptation when little data is available.

3.6 GMM Subspace Modeling

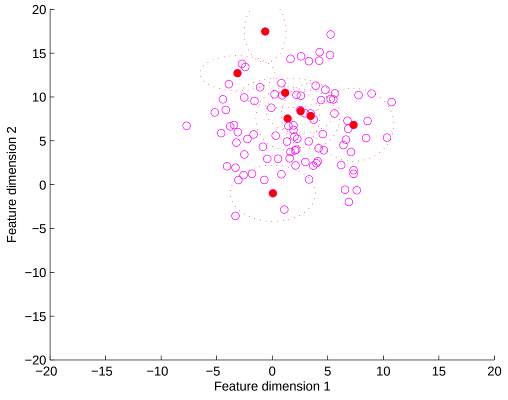
Generally, subspace modeling is a term that refers to representing the parameters of some model by the means of another set of parameters (usually of much lower dimension) to model some selected variability in the original parameter space. This selected variability is given as a set of hyper-parameters for the selected model and tells how the model parameters vary by being trained on different data. The assumption for using these models is that we believe that some parameters of the model are given by useful information (e.g. the speaker information) while other parameters might be rather affected by the noise. As such, we distinguish between the *wanted* variability and the *unwanted* variability.

For better understanding, let us briefly give an SRE example. In the case of SRE, the wanted variability is usually the speaker variability. If our model is large (e.g. supervector of means for thousands of Gaussian components) and the training utterance is relatively short (couple of seconds), then we face the problem of overfitting the model parameters in the enrollment phase. By restricting the model parameter space, we believe that the model is trained more robustly. On the other hand, having trained multiple models for one speaker on different sessions, one could observe that the models differ in some particular subspace, which is given by the unwanted (*inter-session*) variability—generally referred to as *channel*—which can lead to errors in testing. To face these two problems, we could basically enroll the model by adapting only some parameters, and—when testing an utterance—we could adapt some other parameters to compensate for the channel. Or we could essentially combine the two approaches and do both.

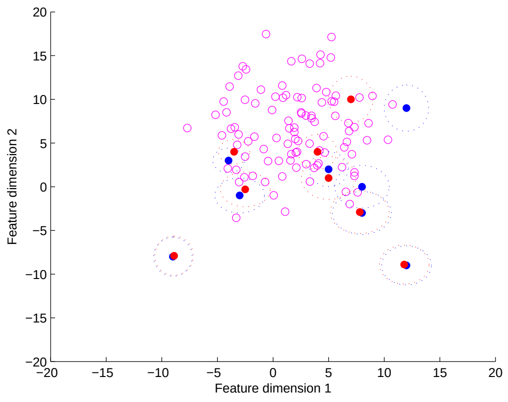
Subspace modeling found its usage in many fields of speech processing, e.g. the subspace GMM [Povey et al., 2011], Cluster Adaptive Training (CAT) [Gales, 1999a], Eigen-Voices [Kuhn et al., 1998] in LVCSR, multinomial channel compensation in LID [Glembek et al., 2008], multinomial subspace models in prosodic



(a) UBM estimation



(b) Model ML estimation



(c) Model MAP estimation

Figure 3.2: GMM as a basic model for speaker recognition

SRE [Kockmann, 2012] etc. In acoustic SRE, the technique was introduced by the means of eigen-voices [Thyes et al., 2000, Kenny et al., 2003], to represent the speaker- dependent mean vector in a low-dimensional space. Since then, many studies were carried out, such as Joint Factor Analysis [Kenny, 2005], eigen-channel adaptation [Brümmer, 2004, Brümmer et al., 2007], and lately the concept of i-vectors [Dehak et al., 2010].

3.6.1 Theoretical Background

Let us look at the supervector of means $\boldsymbol{\mu}$ as a multi-dimensional normal-distributed variable. As such, it lies in a CF-dimensional parameter space. Subspace Modeling, as understood in this work, is modeling the supervector $\boldsymbol{\mu}$ in a linear subspace of the original parameter space with respect to some selected variability. In other words, the aim is to estimate $\boldsymbol{\mu}$ in a low-dimensional space as a linear combination of small number of vector bases which represent the selected variability of model parameters. Mathematically speaking, for an utterance i[4],

$$\underline{\boldsymbol{\mu}}_i = \underline{\mathbf{m}} + \underline{\mathbf{V}}\underline{\mathbf{y}}_i \ , \tag{3.34}$$

where, $\underline{\mathbf{m}}$ represents the offset of the model given by the known variables, $\underline{\mathbf{V}}$ is the (low-rank) matrix defining the sub-space and defines the *hyper-parameters* of the model, and $\underline{\mathbf{y}}_i$ is a hidden variable with standard normal prior, i.e.

$$p(\underline{\mathbf{y}}_i) = \mathcal{N}\left(\underline{\mathbf{y}}_i; \mathbf{0}, \mathbf{1}\right) \ . \tag{3.35}$$

From (3.34) and (3.35) it follows that the prior distribution for $\underline{\boldsymbol{\mu}}_i$ is given as

$$p(\underline{\boldsymbol{\mu}}_i) = \mathcal{N}\left(\underline{\boldsymbol{\mu}}_i; \underline{\mathbf{m}}, \underline{\mathbf{V}}\underline{\mathbf{V}}'\right) \ . \tag{3.36}$$

Similar to the basic GMM, each utterance i is associated with a hidden variable $\underline{\mathbf{y}}_i$. This leads to application of the EM algorithm when estimating the hyper-parameters. As such, we need to be able express the likelihood in terms of the hidden-variable posteriors.

Note that we can decompose the matrix $\underline{\mathbf{V}}$ into C into sub-matrices, each being associated with the corresponding GMM component:

$$\underline{\mathbf{V}} = \begin{bmatrix} \underline{\mathbf{V}}^{(1)} \\ \vdots \\ \underline{\mathbf{V}}^{(C)} \end{bmatrix} , \tag{3.37}$$

3.6.2 The Likelihood Function

The log-likelihood function is based on the general GMM log-likelihood function as defined in Section 3.3. For simplification, the method assumes fixed data alignment [Kenny, 2005]. This lets us represent the log-likelihood by the means of the sufficient statistics. This assumption will be held throughout the rest of the work and we use the UBM for sufficient statistics extraction, as defined in Section 3.2. As was discussed in Section 3.3.1, this assumption implies that the log-likelihood is a lower-bound to the real log-likelihood, which will be denoted by the tilde ˜ symbol. For a single utterance i, let

$$S_i = \langle \mathbf{N}_i, \mathbf{f}_i, \mathbf{S}_i \rangle \tag{3.38}$$

be the collection of the sufficient statistics. The (approximated) log-likelihood is then given by substituting $\boldsymbol{\mu}$ in (3.19) by (3.34), giving

$$\begin{aligned} \log \tilde{p}(S_i | \underline{\mathbf{y}}_i, \underline{\mathbf{V}}) = & \sum_{c=1}^{C} N_i^{(c)} \log \frac{1}{(2\pi)^{F/2} |\boldsymbol{\Sigma}^{(c)}|^{1/2}} \\ & - \frac{1}{2} \mathrm{tr}(\boldsymbol{\Sigma}^{-1} \mathbf{S}_i) \\ & + \underline{\boldsymbol{\mu}}_i{}' \boldsymbol{\Sigma}^{-1} \mathbf{f}_i - \frac{1}{2} \underline{\boldsymbol{\mu}}_i{}' \mathbf{N}_i \boldsymbol{\Sigma}^{-1} \underline{\boldsymbol{\mu}}_i \\ & - \sum_{c=1}^{C} \gamma_i^{(c)} \log \frac{\gamma_i^{(c)}}{w^{(c)}} \end{aligned} \tag{3.39}$$

[4]This concept is common to all subspace techniques studied in this work. This is denoted by the underlined symbols. They can be understood as general symbols which will be substituted by the appropriate variables, depending on the application.

and further

$$\begin{aligned} &= \underline{\mathbf{m}}'\boldsymbol{\Sigma}^{-1}\mathbf{f}_i - \frac{1}{2}\underline{\mathbf{m}}'\mathbf{N}_i\boldsymbol{\Sigma}^{-1}\underline{\mathbf{m}} \\ &\quad + \underline{\mathbf{y}}_i{}'\underline{\mathbf{V}}'\boldsymbol{\Sigma}^{-1}\mathbf{f}_i - \frac{1}{2}\underline{\mathbf{y}}_i{}'\underline{\mathbf{V}}'\mathbf{N}_i\boldsymbol{\Sigma}^{-1}\underline{\mathbf{V}}\underline{\mathbf{y}}_i \\ &\quad + \text{const} . \end{aligned} \tag{3.40}$$

3.6.3 Posterior of the Hidden Variables

Let us assume for this moment that the hyper-parameters $\underline{\mathbf{V}}$ are known. Since the likelihood of the data—as given by (3.39)—is Gaussian, and the prior for the hidden variables is standard normal, then the posterior for the hidden variables is Gaussian [Bishop, 2006]:

$$p(\underline{\mathbf{y}}_i|S_i, \underline{\mathbf{V}}) = \mathcal{N}\left(\underline{\mathbf{y}}_i; \hat{\underline{\mathbf{y}}}_i, \underline{\mathbf{L}}_i^{-1}\right) , \tag{3.41}$$

with the mean $\hat{\underline{\mathbf{y}}}$ given as

$$\hat{\underline{\mathbf{y}}}_i = \underline{\mathbf{L}}_i^{-1}\underline{\mathbf{V}}'\boldsymbol{\Sigma}^{-1}\overline{\mathbf{f}}_i , \tag{3.42}$$

where $\overline{\mathbf{f}}_i$ is defined as shifted fist-order statistics:

$$\overline{\mathbf{f}}_i = \mathbf{f}_i - \mathbf{N}_i\underline{\mathbf{m}} , \tag{3.43}$$

and $\underline{\mathbf{L}}_i$ is the precision matrix of the posterior distribution computed as

$$\underline{\mathbf{L}}_i = \mathbf{I} + \underline{\mathbf{V}}'\mathbf{N}_i\boldsymbol{\Sigma}^{-1}\underline{\mathbf{V}} . \tag{3.44}$$

3.6.4 Hyper-Parameter Estimation

The objective for the training is to maximize the likelihood of the training data $p(S_i|\underline{\mathbf{V}}, \underline{\mathbf{y}}_i)$, which is a marginalization of (3.39) over the hidden variables (as discussed in Appendix A.1). We can make use of the Bayes' rule and express the marginal using the probabilities which we already know:

$$p(S_i|\underline{\mathbf{V}}) = \frac{p\left(S_i|\underline{\mathbf{y}}_i, \underline{\mathbf{V}}\right) p(\underline{\mathbf{y}}_i|\underline{\mathbf{V}})}{p\left(\underline{\mathbf{y}}_i|S_i, \underline{\mathbf{V}}\right)} , \tag{3.45}$$

where $p(\underline{\mathbf{y}}_i|\underline{\mathbf{V}})$ is constant and so

$$p(S_i|\underline{\mathbf{V}}) \propto \frac{p\left(S_i|\underline{\mathbf{y}}_i, \underline{\mathbf{V}}\right)}{p\left(\underline{\mathbf{y}}_i|S_i, \underline{\mathbf{V}}\right)} . \tag{3.46}$$

Note that the formula is true for any value of $\underline{\mathbf{y}}_i$. The numerator and denominator for (3.46) are computed using (3.40) and (3.41), respectively, and for convenience, we choose $\underline{\mathbf{y}}_i = \mathbf{0}$. The logarithm of (3.46) then becomes

$$\log p\left(S_i|\underline{\mathbf{V}}\right) = \underline{\mathbf{m}}'\boldsymbol{\Sigma}^{-1}\mathbf{f}_i - \frac{1}{2}\underline{\mathbf{m}}'\mathbf{N}_i\boldsymbol{\Sigma}^{-1}\underline{\mathbf{m}} - \log|\underline{\mathbf{L}}_i| + \frac{1}{2}\hat{\underline{\mathbf{y}}}_i{}'\underline{\mathbf{L}}\hat{\underline{\mathbf{y}}}_i + \text{const} . \tag{3.47}$$

Leaving out the constant term, the objective for estimating $\underline{\mathbf{V}}$ is given as

$$\mathcal{L}(\underline{\mathbf{V}}) = \sum_i \underline{\mathbf{m}}'\boldsymbol{\Sigma}^{-1}\mathbf{f}_i - \frac{1}{2}\underline{\mathbf{m}}'\mathbf{N}_i\boldsymbol{\Sigma}^{-1}\underline{\mathbf{m}} - \log|\underline{\mathbf{L}}_i| + \frac{1}{2}\hat{\underline{\mathbf{y}}}_i{}'\underline{\mathbf{L}}\hat{\underline{\mathbf{y}}}_i . \tag{3.48}$$

Expectation Maximization

Similar to the GMM training, the EM algorithm for the hyper-parameter estimation comprises two steps that are repeated iteratively [Brümmer, 2009b]. In the E step, as described in Appendix A.2, given the initial parameters $\underline{\mathbf{V}}_0$, the auxiliary function is constructed as an expected value of the complete log-likelihood (constructed using (3.39) and (3.35)) over the hidden-variable posteriors (3.41):

$$
\begin{aligned}
\mathcal{Q}(\underline{\mathbf{V}}, \underline{\mathbf{V}}_0) &= \sum_i \mathrm{E}\left[\log \frac{p(S_i, \underline{\mathbf{y}}_i | \underline{\mathbf{V}})}{p\left(\underline{\mathbf{y}}_i | S_i, \underline{\mathbf{V}}_0\right)}\right] && (3.49)\\
&= \sum_i \mathrm{E}\left[\log p(S_i | \underline{\mathbf{y}}_i, \underline{\mathbf{V}}) + \mathrm{const}\right] && (3.50)\\
&= \sum_i \mathrm{E}\left[\underline{\mathbf{m}}'\boldsymbol{\Sigma}^{-1}\mathbf{f}_i - \frac{1}{2}\underline{\mathbf{m}}'\mathbf{N}_i\boldsymbol{\Sigma}^{-1}\underline{\mathbf{m}} + \underline{\mathbf{y}}_i{}'\underline{\mathbf{V}}'\boldsymbol{\Sigma}^{-1}\mathbf{f}_i - \frac{1}{2}\underline{\mathbf{y}}_i{}'\underline{\mathbf{V}}'\mathbf{N}_i\boldsymbol{\Sigma}^{-1}\underline{\mathbf{V}}\underline{\mathbf{y}}_i + \mathrm{const}\right] && (3.51)\\
&= \sum_i \mathrm{E}\left[\underline{\mathbf{y}}_i{}'\underline{\mathbf{V}}'\boldsymbol{\Sigma}^{-1}\mathbf{f}_i - \frac{1}{2}\underline{\mathbf{y}}_i{}'\underline{\mathbf{V}}'\mathbf{N}_i\boldsymbol{\Sigma}^{-1}\underline{\mathbf{V}}\underline{\mathbf{y}}_i\right] + \mathrm{const} && (3.52)\\
&= \sum_i \mathrm{tr}\left(\underline{\mathbf{C}}_i\underline{\mathbf{V}}'\right) - \frac{1}{2}\mathrm{tr}\left(\mathbf{A}_i\underline{\mathbf{V}}'\mathbf{N}_i\underline{\mathbf{V}}\right) + \mathrm{const} && (3.53)
\end{aligned}
$$

where the expectations are over $p\left(\underline{\mathbf{y}}_i | S_i, \underline{\mathbf{V}}_0\right)$, and

$$
\begin{aligned}
\underline{\mathbf{C}}_i &= \mathbf{f}_i\hat{\underline{\mathbf{y}}}_i{}' \\
\underline{\mathbf{A}}_i &= \hat{\underline{\mathbf{y}}}_i\hat{\underline{\mathbf{y}}}_i{}' + \underline{\mathbf{L}}_i \; .
\end{aligned}
\tag{3.54}
$$

We can derive (3.53) with respect to each sub-matrix $\underline{\mathbf{V}}^{(c)}$ of $\underline{\mathbf{V}}$ as defined by (3.37):

$$
\frac{\mathrm{d}\mathcal{Q}(\underline{\mathbf{V}}, \underline{\mathbf{V}}_0)}{\mathrm{d}\underline{\mathbf{V}}^{(c)}} = \sum_i \underline{\mathbf{C}}_i^{(c)\prime} - \left(\sum_i n_i^{(c)}\underline{\mathbf{A}}_i^{(c)}\right)\underline{\mathbf{V}}^{(c)\prime} \tag{3.55}
$$

Setting the derivative to zero, yields a closed-form solution for computing the hyperparameters:

$$
\underline{\mathbf{V}}^{(c)} = \underline{\mathbf{C}}^{(c)}\underline{\mathbf{A}}^{(c)^{-1}} \; , \tag{3.56}
$$

where

$$
\begin{aligned}
\underline{\mathbf{C}}^{(c)} &= \sum_i \underline{\mathbf{C}}_i^{(c)} \\
\underline{\mathbf{A}}^{(c)} &= \sum_i n_i^{(c)}\underline{\mathbf{A}}_i^{(c)}
\end{aligned}
\tag{3.57}
$$

are the accumulators that are collected during the E-step.

Minimum Divergence

The Minimum Divergence (MD) step is complementary to the M-step of the EM algorithm (see Appendix A.2 for explanation). The task is to find new hyper-parameters $\underline{\mathbf{V}}$ from initial parameters $\underline{\mathbf{V}}_0$ to minimize the divergence from the hidden variable posterior to its prior. In simple words, the prior for $\underline{\mathbf{y}}$ is assumed to be standard normal, and we should observe this when estimating the (Gaussian) distribution of $\hat{\underline{\mathbf{y}}}$ over the training set. However, we can find that it is not so. MD aims at updating this prior and then rotating matrix $\underline{\mathbf{V}}$ in such a way that the estimates $\hat{\underline{\mathbf{y}}}_i$ are standard normal distributed over the training set again. Note that in this case, MD does not increase the log-likelihood of the training data, rather it yields faster EM convergence by updating the prior of the hidden variables. Generally, the algorithm also applies to updating $\underline{\mathbf{m}}$ from $\underline{\mathbf{m}}_0$, but as was mentioned earlier, this step is skipped in this work.

Let

$$\mathbf{Y} = \left[\underline{\mathbf{y}}_1 \, \underline{\mathbf{y}}_2 \cdots\right] \tag{3.58}$$

denote all hidden variables for all data, and

$$\mathcal{D} = \langle d_1, d_2, \cdots \rangle \tag{3.59}$$

denote the collection of all data. Let the prior be temporarily non-standard. Then, the prior and posterior distributions of all data are given, respectively, as

$$p\left(\mathbf{Y}|\bar{\mathbf{y}}, \bar{\mathbf{P}}\right) = \prod_i \mathcal{N}\left(\underline{\mathbf{y}}_i; \bar{\mathbf{y}}, \bar{\mathbf{P}}^{-1}\right) \tag{3.60}$$

$$p\left(\mathbf{Y}|\mathcal{D}, \underline{\mathbf{V}}_0\right) = \prod_i \mathcal{N}\left(\underline{\mathbf{y}}_i; \hat{\underline{\mathbf{y}}}_i, \mathbf{L}_i^{-1}\right) \ , \tag{3.61}$$

where $\bar{\mathbf{y}}$ and $\bar{\mathbf{P}}$ are the mean and precision of the new prior, respectively. The divergence is given as

$$\mathrm{D_{KL}}\left(p\left(\mathbf{Y}|\mathcal{D}, \underline{\mathbf{V}}_0\right) \| \, p\left(\mathbf{Y}|\bar{\mathbf{y}}, \bar{\mathbf{P}}\right)\right) = \mathrm{E}_{p(\mathbf{Y}|\mathcal{D}, \underline{\mathbf{V}}_0)}\left[\log \frac{p\left(\mathbf{Y}|\mathcal{D}, \underline{\mathbf{V}}_0\right)}{p\left(\mathbf{Y}|\bar{\mathbf{y}}, \bar{\mathbf{P}}\right)}\right] \ . \tag{3.62}$$

Deriving (3.62) w.r.t. $\bar{\mathbf{y}}$ and $\mathbf{P}$, and setting to zero, the parameters are computed as

$$\bar{\mathbf{y}} = \frac{1}{N}\sum_{i=1}^{N} \hat{\underline{\mathbf{y}}}_i \tag{3.63}$$

$$\mathbf{P}^{-1} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{L}_i + (\hat{\underline{\mathbf{y}}}_i - \bar{\mathbf{y}})(\hat{\underline{\mathbf{y}}}_i - \bar{\mathbf{y}})' \tag{3.64}$$

By normalizing the hidden variables using an affine transform

$$\phi(\underline{\mathbf{y}}) = \mathbf{J}^{-1}(\underline{\mathbf{y}} - \bar{\mathbf{y}}) \ , \tag{3.65}$$

where $\mathbf{J}$ is a symmetric decomposition of $\mathbf{P}^{-1}$ so that

$$\mathbf{J}\mathbf{J}' = \mathbf{P}^{-1} \tag{3.66}$$

(such as Cholesky decomposition), we make the prior standard normal again. The task is to find new parameters $\underline{\mathbf{V}}$ (and generally $\underline{\mathbf{m}}$) such that

$$\underline{\mathbf{m}} + \underline{\mathbf{V}}\underline{\mathbf{y}} = \underline{\mathbf{m}}_0 + \underline{\mathbf{V}}_0\phi(\underline{\mathbf{y}}) \tag{3.67}$$

$$= \underline{\mathbf{V}}_0\mathbf{J}\underline{\mathbf{y}} + (\underline{\mathbf{V}}_0\bar{\underline{\mathbf{y}}} + \underline{\mathbf{m}}_0) \ . \tag{3.68}$$

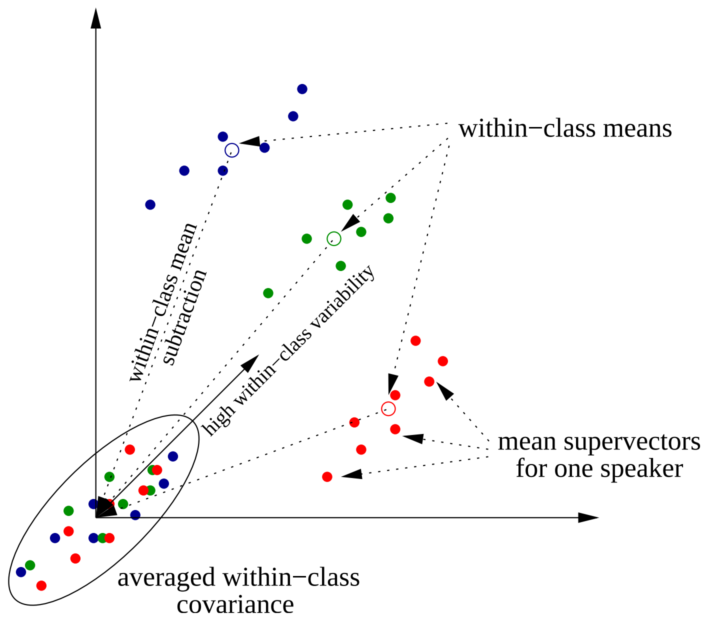

within−class means
within−class mean subtraction
high within−class variability
mean supervectors for one speaker
averaged within−class covariance

Figure 3.3: Alternative estimation of the subspace-GMM hyper-parameters via PCA

The general update formula is then given as

$$\underline{\mathbf{V}} = \underline{\mathbf{V}}_0 \mathbf{J} \tag{3.69}$$

$$\underline{\mathbf{m}} = \underline{\mathbf{V}}_0 \bar{\underline{\mathbf{y}}} + \underline{\mathbf{m}}_0 \ , \tag{3.70}$$

where re-estimation of $\underline{\mathbf{m}}$ is skipped.

Alternative Estimation via PCA

Simplified estimation has been proposed in [Brümmer, 2004]. In this case, the estimate is given by R eigenvectors of the average within-class covariance matrix, where each class is represented by the MAP-adapted mean supervectors estimated on different segments spoken by the same speaker. Figure 3.3 depicts the computation of the averaged within-class covariance matrix.

3.6.5 The Channel-Compensation Example

Throughout the years of research, various techniques for dealing with the so-called *inter-session* variability have been introduced. Figure 3.4 shows the effect of channel mismatch for an exemplary system evaluated on the NIST2008 Interview condition, where two different microphones were used for enrolling and test.[5] At the feature level, there is a feature normalization[6] (see e.g. [Openshaw and Masan, 1994]) already mentioned in Section 1.1.2.

[5]The figure is a proprietary image of Lukáš Burget and was copied with his kind permission.

[6]Feature normalization is often referred to as cepstral mean and variance normalization (CMN, CVN) due to the fact, that cepstral features are used most often.

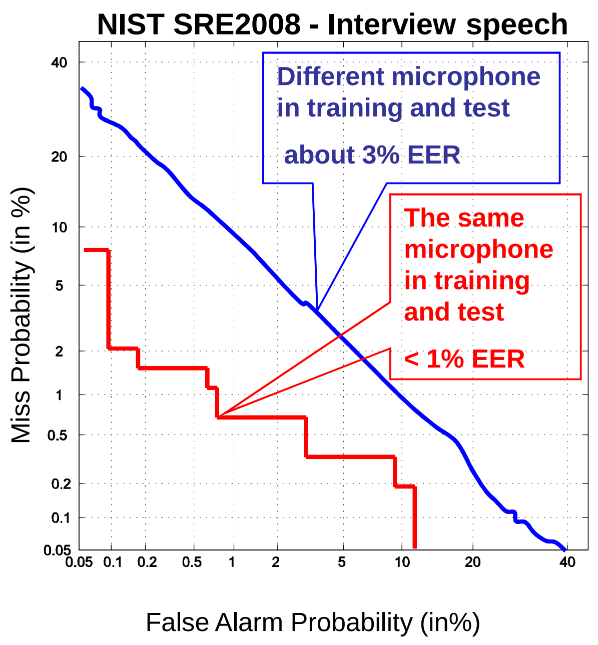
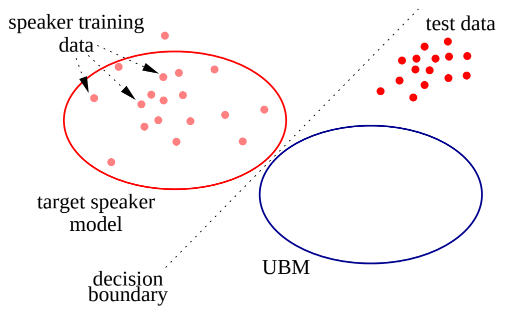


Figure 3.4: Example of channel mismatch effect. The red DET curve shows a system in which the same microphone was used for speaker model training and for test. The blue curve, however, shows what happens when two different microphones are used.

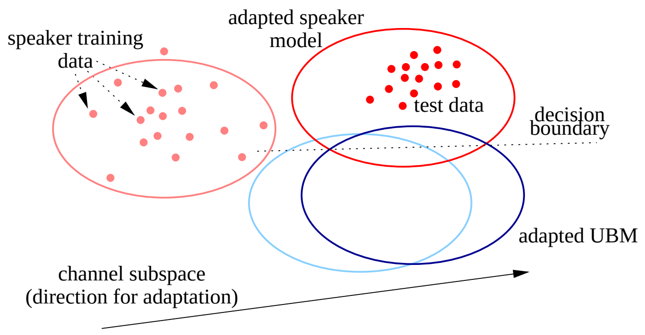
Another successful technique used to deal with the channel is *feature mapping*, introduced in [Reynolds, 2003]. The technique is based on mapping feature vectors into a channel-independent space. The mapping is learnt from a set of channel-dependent models. The principle is as follows: given an input utterance, the most likely channel-dependent model is first detected and then each feature vector in the utterance is mapped to the channel independent space based on its top-1 decoded Gaussian in the channel dependent GMM. The author claims a "significant" improvement on the NIST SRE2002 task (around 30% relative improvement on the EER as read from the DET plots). This technique suffers from relying on a channel detector which is basically a closed-set multi-class classifier. Therefore, if an input utterance comes from an unseen channel, the technique might fail.

The problem of channel compensation has recently been addressed from the point of subspace modeling. The scenario is depicted in Figure 3.5.a. The figure shows a situation when the test data come from the target speaker, but the log-likelihood ratio is clearly negative, i.e. the decision is in favor of the UBM. The data mismatch is caused by the channel difference. In the GMM mean space, one can identify a subspace of *eigenchannels*[7] that covers the unwanted variability. The solution to this issue is to use the test data to adapt the models in the eigenchannel space, as shown in Figure 3.5.b. As opposed to the standard MAP adaptation as described in Section 3.5, the eigenchannel adaptation (both

[7]The term eigenchannel—as used in SRE—was adopted from [Kenny and Dumouchel, 2004]. It was introduced to the NIST SRE in 2004 by SDV [Brümmer, 2004], revisited by Kenny and Vogt [Vogt et al., 2005] in SRE 2005, and again by various sites in the following SRE's.



(a) Detection with no channel adaptation



(b) Detection with channel-adapted models

Figure 3.5: Channel adaptation

of the speaker model and the UBM) is restricted to the eigenchannel subspace, only.

Mathematically, the (eigenchannel-adapted) model $\hat{\boldsymbol{\mu}}$ is given as

$$\hat{\boldsymbol{\mu}} = \boldsymbol{\mu} + \mathbf{c}_d, \tag{3.71}$$

where $\boldsymbol{\mu}$ is the channel-independent model and $\mathbf{c}_d$ is the supervector of channel-shift for the input d, and it is defined as:

$$\mathbf{c}_d = \mathbf{U}\mathbf{x}_d, \tag{3.72}$$

where $\mathbf{U}$ is a $CF \times R$ matrix whose columns define the eigenchannel subspace, and $\mathbf{x}_d$ is a vector of *channel factors* which are dependent on the input d. Imposing a standard normal prior on $\mathbf{x}$ we can use the framework of GMM subspace modeling as described in previous sections and use the following assignment to apply the framework to channel adaptation:

$$\underline{\mathbf{y}} := \mathbf{x} \tag{3.73}$$

$$\underline{\mathbf{V}} := \mathbf{U} \tag{3.74}$$

$$\underline{\mathbf{m}} := \boldsymbol{\mu}\ . \tag{3.75}$$

Figure 3.6 shows an exemplary result of one of our NIST2005 systems.[8] We used the alternative PCA technique to estimate the channel subspace, as described in Section 3.6.4. We see the superior performance of eigen-channel adaptation and depicts the strength of GMM subspace modeling. For a thorough analysis on the NIST data, see e.g. [Burget et al., 2007].

[8]The figure is a proprietary image of Lukáš Burget and was copied with his kind permission.

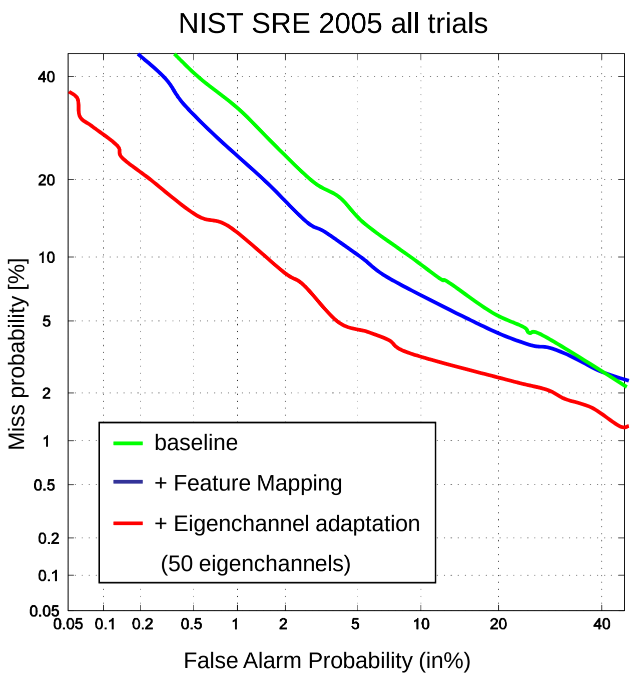


Figure 3.6: Comparison of the different channel compensation techniques on the complete NIST2005 data. We see the superior performance of the eigen-channel adaptation.

3.7 Claims of the Thesis Revisited

Let us summarize the claims of the thesis, as stated in Section 1.3.1, in terms of the knowledge that was presented in this chapter:

- Claim 1 will be presented in terms of the different likelihood computation as given in Section 3.3.
- Claim 2 will be presented in terms of different assumptions for computing the posterior probability of the subspace hiden-variable, as given in Section 3.6.3.
- Claim 3 is based on the knowledge from Claim 2 and on the hyper-parameter estimation as given in Section 3.6.4.
- Claim 4 is based on the study of discriminatively training the hyper-parameters of a GMM-subspace model (Section 3.6) with the C_{llr} objective function as described in Section 2.3.5.

Chapter 4

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. 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.

4.1 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 continuous feature space of dimension F. The GMM for a target speaker is obtained by adapting the Universal Background Model (UBM) mean parameters. Similar to the eigen-channel adaptation, in Joint Factor Analysis [Kenny et al., 2007], the assumption is that a speaker- and channel-dependent supervector of means $\boldsymbol{\mu}$ can be decomposed into a sum of two supervectors: a speaker supervector $\mathbf{s}$ and a channel supervector $\mathbf{c}$

$$\boldsymbol{\mu} = \mathbf{s} + \mathbf{c}, \tag{4.1}$$

where $\mathbf{s}$ and $\mathbf{c}$ are normally distributed. The decomposition is shown in Figure 4.1. As opposed to the eigen-channel adaptation, not only the channel is represented in a low-dimensional subspace, but also the speaker. 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 (4.1) is modeled by assuming that if $\mathbf{s}$ is the speaker supervector for a randomly chosen speaker then

$$\mathbf{s} = \mathbf{m} + \mathbf{V}\mathbf{y} + \mathbf{D}\mathbf{z}, \tag{4.2}$$

where $\mathbf{m}$ is the speaker and channel independent supervector (usually the supervector of UBM means), $\mathbf{D}$ is a diagonal matrix, $\mathbf{V}$ is a rectangular matrix of low rank and $\mathbf{y}$ and $\mathbf{z}$ are hidden variables having standard normal prior distributions. In other words, $\mathbf{s}$ is

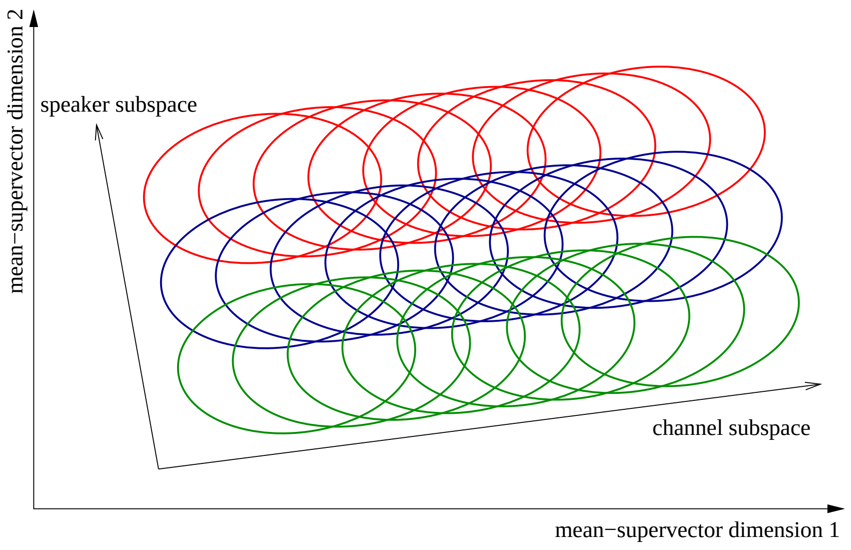


Figure 4.1: GMM probability density function for 2-dimensional data

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

$$\mathbf{c} = \mathbf{U}\mathbf{x}, \tag{4.3}$$

where $\mathbf{U}$ is a rectangular matrix of low rank (known as eigenchannel matrix), $\mathbf{x}$ is again a hidden variable with standard normal prior 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 referred to as channel factors. Figure 4.2 shows the complete decomposition of the supervector space.

4.1.1 Model Training

Training scheme of the hyper-parameters has undergone a series of simplifications since the very first experiments (see e.g. [Kenny and Dumouchel, 2004, Kenny et al., 2005]). In the beginning, all matrices were trained jointly resulting in a slow process (for theory see [Kenny, 2005]). For large-scale models, this approach was very difficult to use. By further simplifications (e.g. [Kenny et al., 2005, Kenny et al., 2008, Burget et al., 2009]), a decoupled training scheme has been introduced, i.e. the matrices of hyper-parameters were trained one after another with the hidden variables fixed to their point estimate. This allows for adapting the general GMM subspace modeling technique and using it to compute the posteriors of the hidden variables (and their mean as the point estimate), and to train the hyper-parameters of the model. Below is a training scheme that was adopted for the experiments in this work and that has proved to work very well and was successfully used during various NIST SRE speaker evaluations.

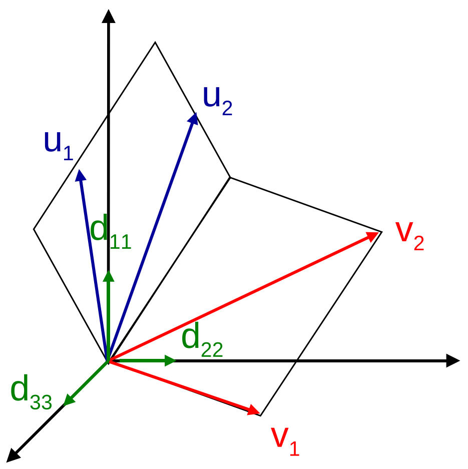


Figure 4.2: GMM probability density function for 2-dimensional data

- **Step 1**: Eigen-voice matrix $\mathbf{V}$ is estimated assuming that $\mathbf{U}$ and $\mathbf{D}$ are set to $\mathbf{0}$:

$$\begin{aligned} \underline{\mathbf{y}} &:= \mathbf{y} \\ \underline{\mathbf{V}} &:= \mathbf{V} \\ \underline{\mathbf{m}} &:= \mathbf{m} \end{aligned} , \tag{4.4}$$

- **Step 2**: Speaker factors $\mathbf{y}$ are computed based on the speaker labels, i.e. all utterances from one speaker are used to (robustly) estimate the speaker factor which is then distributed among the associated utterances. Matrix $\mathbf{D}$ is still considered to be $\mathbf{0}$.

$$\begin{aligned} \underline{\mathbf{y}} &:= \mathbf{x} \\ \underline{\mathbf{V}} &:= \mathbf{U} \\ \underline{\mathbf{m}} &:= \mathbf{m} + \mathbf{V}\mathbf{y} \end{aligned} , \tag{4.5}$$

- **Step 3**: Speaker factors $\mathbf{y}$ are computed as in the previous step and channel factors are estimated for each utterance separately. Then the residual $\mathbf{D}$ matrix is estimated:

$$\begin{aligned} \underline{\mathbf{y}} &:= \mathbf{z} \\ \underline{\mathbf{V}} &:= \mathbf{D} \\ \underline{\mathbf{m}} &:= \mathbf{m} + \mathbf{V}\mathbf{y} + \mathbf{U}\mathbf{x} \end{aligned} , \tag{4.6}$$

Depending on the training phase, estimation of the parameters can be now generally described using the terms $\underline{\mathbf{y}}$, $\underline{\mathbf{V}}$, and $\underline{\mathbf{m}}$. Each of the steps defined above comprises application of the Expectation Maximization algorithm and the Minimum Divergence step as described in Section 3.6.4. Kenny also suggests training of the $\mathbf{m}$ parameter, however, following the work of [Burget et al., 2009], it is set to the mean supervector of the UBM.

Later in Section 6.2 we will show how the eigen-voices matrix can be optimized in a discriminative framework. Cross-entropy, as defined early in Section 2.3.5, will be used as the optimization criterion. The optimization will be based on one of the scoring methods presented in the following section.

4.2 Comparison of Scoring Methods

Many sites used JFA in the past NIST evaluations, however they report their results using different scoring methods ([Kenny et al., 2007], [Vair et al., 2007], [Brümmer et al., 2007]). The aim of this work was to compare these techniques in terms of speed and performance. It is necessary to point out that the speed test was performed from a scientist's point of view, i.e. the process starts at the time of loading the necessary data units and ends at generating a set of scores for the given test set. The data units differ among the described methods and significantly determine the speed of the algorithms. To be more specific, the slowest method loads the whole feature files, however the fast methods only loads sufficient statistics which already saves a lot of time.

Scoring in JFA is performed in the asymmetrical way, i.e. data d_{s} is used to train a model for speaker s and log-likelihood is computed using test data d_{t}. In the Bayesian framework, the speaker model is given by the posterior distribution of $\mathbf{y}_s$ and $\mathbf{z}_s$[1], where

$$\begin{aligned} p(\mathbf{y}_s|d_{\mathrm{s}}) &= \mathcal{N}(\mathbf{y}_s;\hat{\mathbf{y}}_s,\mathbf{L}_{y,s}^{-1}) \\ p(\mathbf{z}_s|d_{\mathrm{s}}) &= \mathcal{N}(\mathbf{z}_s;\hat{\mathbf{z}}_s,\mathbf{L}_{z,s}^{-1}) \ , \end{aligned} \tag{4.7}$$

where the hat version of a symbol represents the mean of the distribution, and $\mathbf{L}_{\cdot,s}$ represents the precision matrix. The Bayesian likelihood of test utterance d_{t} 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]:

$$p(d_{\mathrm{t}}|d_{\mathrm{s}}) = \iiint p(d_{\mathrm{t}}|\mathbf{x},\mathbf{y},\mathbf{z})\mathcal{N}(\mathbf{x};\mathbf{0},\mathbf{I})p(\mathbf{y}|d_{\mathrm{s}})p(\mathbf{z}|d_{\mathrm{s}})\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{z} \tag{4.8}$$

In [Kenny et al., 2007], it was later shown, that using mere MAP point estimates of $\mathbf{y}$ and $\mathbf{z}$ is sufficient. This means that we can treat the factors as known variables and we can set them to the means of the posterior distribution. Still, integration over the prior distribution of $\mathbf{x}$ was performed as given by [Kenny et al., 2007, Equation (13)]:

$$p(d_{\mathrm{t}}|\mathbf{s}) = \int p(d_{\mathrm{t}}|\mathbf{s},\mathbf{x})\mathcal{N}(\mathbf{x};\mathbf{0},\mathbf{I})\mathrm{d}\mathbf{x} \ . \tag{4.9}$$

It is further shown 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 d_{t}.

[1]Note that subscript $_s$ is added to the factor variable to denote the connection with the enrolled speaker

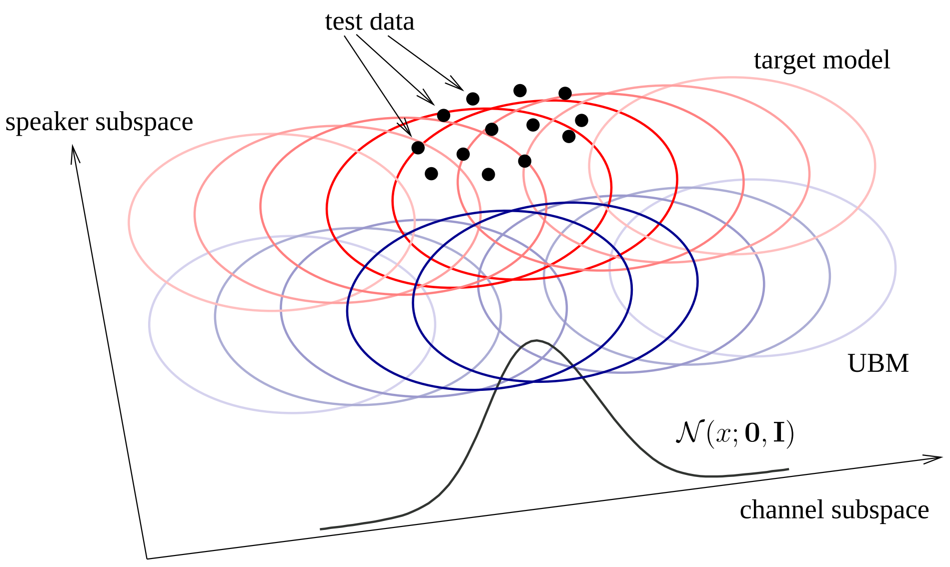


Figure 4.3: Evaluation of log-likelihood with integration over the channel subspace. The transparency depicts the probability $p(d_{\mathrm{t}}|\mathbf{s},\mathbf{x})\mathcal{N}(\mathbf{x};\mathbf{0},\mathbf{I})$

4.2.1 Frame by Frame

Frame-by-Frame is based on a full GMM log-likelihood evaluation. The log-likelihood of utterance d_{t} and model $\mathbf{s}$ is computed as an average frame log-likelihood [2]. It is practically infeasible to integrate out the channel, therefore MAP point estimate of $\mathbf{x}$ is used. The formula is as follows

$$\log p(d_{\mathrm{t}}|\mathbf{s}) = \sum_{t=1}^{T} \log \sum_{c=1}^{C} w^{(c)} \mathcal{N}\left(\mathbf{o}_t; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}\right), \tag{4.10}$$

where $\mathbf{o}_t$ is the feature vector at frame t, T is the length (in frames) for utterance d_{t}, 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.

4.2.2 Integrating over Channel Distribution

This approach is based on evaluating an objective function as given by (4.9). The situation is depicted in Figure 4.3. As was said in the previous paragraph, it would be difficult to evaluate this formula in the frame-by-frame strategy. However, (4.10) can be approximated by using fixed alignment of frames to Gaussians, as was described early in Section 3.3.1. 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 (4.9). The closed

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

form (logarithmic) solution is then given as:

$$\begin{aligned}\log \tilde{p}(d_{\mathrm{t}}|\mathbf{s}) = &\sum_{c=1}^{C} N^{(c)} \log \frac{1}{(2\pi)^{F/2}|\mathbf{\Sigma}^{(c)}|^{1/2}} \\ &- \frac{1}{2}\operatorname{tr}(\mathbf{\Sigma}^{-1}\mathbf{S}_s) - \frac{1}{2}\log|\mathbf{L}| \\ &+ \frac{1}{2}\left\|\mathbf{L}^{-1/2}\mathbf{U}'\mathbf{\Sigma}^{-1}\mathbf{f}_s\right\|^2 \ ,\end{aligned} \tag{4.11}$$

where for the first term, C is the number of Gaussians, $N^{(c)}$ are the coefficients of the zero-order statistics, F is the feature vector size, $\mathbf{\Sigma}^{(c)}$ is covariance matrix for Gaussian c. This term 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 (4.11), $\mathbf{S}_s$ is the second order moment of utterance d_{t} around speaker s given as

$$\mathbf{S}_s = \mathbf{S} - 2\operatorname{diag}(\mathbf{f}\mathbf{s}') + \operatorname{diag}(\mathbf{N}\mathbf{s}\mathbf{s}'), \tag{4.12}$$

where $\mathbf{S}$ is the super-matrix of second-order statistics—it is independent of the speaker, thus will cancel out in the LLR computation (note that this was the only place where second order statistics appeared, therefore, they are not needed for scoring). $\mathbf{f}$ and $\mathbf{N}$ are the first- and zero- order statistics, respectively.

The $\mathbf{L}$ in the third term of (4.11) is the precision of the posterior distribution of the channel factor, given as

$$\mathbf{L} = \mathbf{I} + \mathbf{U}'\mathbf{\Sigma}^{-1}\mathbf{N}\mathbf{U}, \tag{4.13}$$

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 (4.11), let $\mathbf{L}^{1/2}$ be a lower triangular matrix, such that

$$\mathbf{L} = \mathbf{L}^{1/2}\mathbf{L}^{1/2'} \tag{4.14}$$

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

As was said, terms one and three in (4.11), and second order statistics $\mathbf{S}$ in (4.12) will cancel out. Then the formula for the score is given as

$$\begin{aligned}Q_{\mathrm{int}}(d_{\mathrm{t}}|\mathbf{s}) = &\operatorname{tr}\left(\mathbf{\Sigma}^{-1}\operatorname{diag}(\mathbf{f}\mathbf{s}')\right) \\ &+ \frac{1}{2}\operatorname{tr}(\mathbf{\Sigma}^{-1}\operatorname{diag}(\mathbf{N}\mathbf{s}\mathbf{s}')) \\ &+ \frac{1}{2}\|\mathbf{L}^{-1/2}\mathbf{U}'\mathbf{\Sigma}^{-1}\mathbf{f}_{\mathbf{s}}\|^2 \ .\end{aligned} \tag{4.15}$$

4.2.3 Channel Point Estimate

This function is based on directly evaluating the log-likelihood ratio using (3.39). This way, there is no need for integrating over the whole distribution of $\mathbf{x}$, and only its point

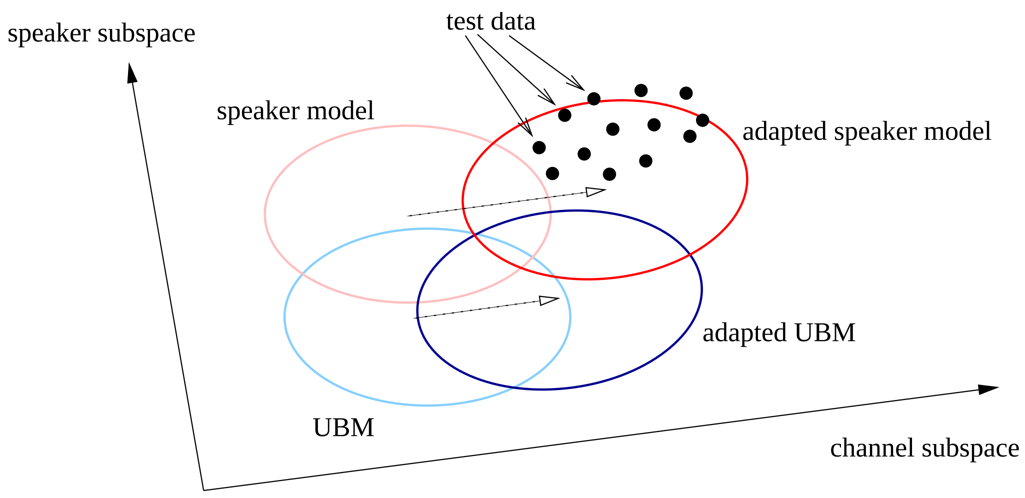


Figure 4.4: Illustration of the "Channel point estimate" technique. Both the speaker model and the UBM are adapted separately.

estimate is taken for LLR computation. In (3.39), the first and second terms cancel out in LLR computation, leading to scoring function

$$Q_{\mathrm{x}}(d_{\mathrm{t}}|\mathbf{s},\mathbf{x}) = \boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\mathbf{f} + \frac{1}{2}\boldsymbol{\mu}'\mathbf{N}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \ , \tag{4.16}$$

hence

$$\mathrm{LLR}_{\mathrm{x}}(d_{\mathrm{t}}|\mathbf{s}) = Q_{\mathrm{x}}(d_{\mathrm{t}}|\mathbf{s},\mathbf{x}_s) - Q_{\mathrm{x}}(d_{\mathrm{t}}|\mathrm{UBM},\mathbf{x}_{\mathrm{UBM}}), \tag{4.17}$$

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

4.2.4 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[3]. Therefore, the $\mathbf{x}$ factor for utterance d_{t} is estimated using the UBM and then used for scoring. Formally written:

$$\mathrm{LLR}_{\mathrm{LPT}}(d_{\mathrm{t}}|\mathbf{s}) = Q_{\mathrm{x}}(d_{\mathrm{t}}|\mathbf{s},\mathbf{x}_{\mathrm{UBM}}) - Q_{\mathrm{x}}(d_{\mathrm{t}}|\mathrm{UBM},\mathbf{x}_{\mathrm{UBM}}) \ . \tag{4.18}$$

Note, that when computing the LLR, $\mathbf{Ux}$ in the linear term of (3.39) will cancel out, leaving the compensation to the quadratic term of (3.39). The situation is depicted in Figure 4.5.

4.2.5 Linear Scoring

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

$$\mathbf{m_c} = \mathbf{m} + \mathbf{c} \ . \tag{4.19}$$

[3]The authors identified themselves under abbreviation LPT as for "Loquendo–Politecnico di Torino", therefore I will refer to this approach as to LPT assumption

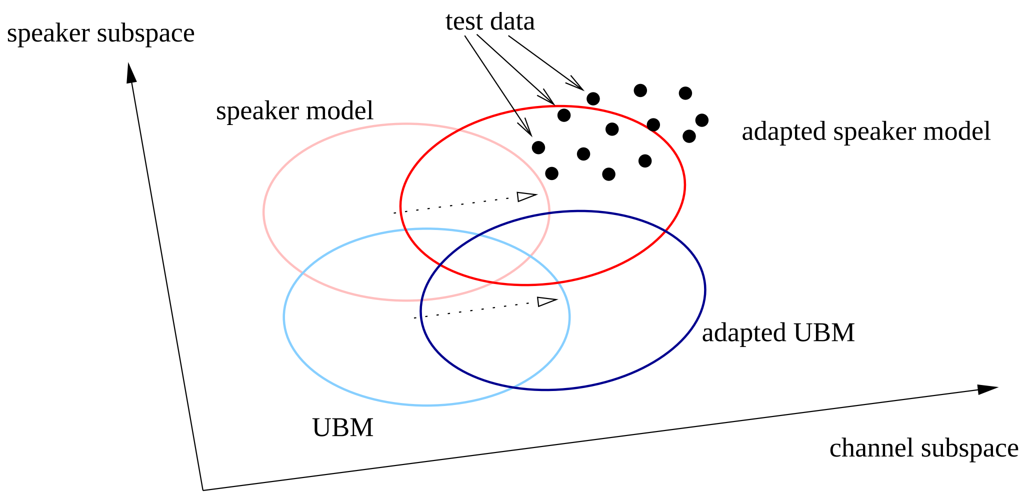


Figure 4.5: Illustration of the "LPT" assumption. The UBM defines the channel adaptation also for the speaker model.

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

$$\bar{\boldsymbol{\mu}} = \boldsymbol{\mu} - \mathbf{m_c} \tag{4.20}$$
$$\bar{\mathbf{f}} = \mathbf{f} - \mathbf{N}\mathbf{m_c} \ . \tag{4.21}$$

Eq. (4.16) can now be rewritten to

$$\begin{aligned} Q_{\text{xmod}}(d_{\text{t}}|\bar{\boldsymbol{\mu}}, \mathbf{x}) = \bar{\boldsymbol{\mu}}'\boldsymbol{\Sigma}^{-1}\bar{\mathbf{f}} \\ + \frac{1}{2}\bar{\boldsymbol{\mu}}'\mathbf{N}\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{\mu}} \ . \end{aligned} \tag{4.22}$$

When approximating (4.22) by the first order Taylor series (as a function of $\bar{\boldsymbol{\mu}}$), only the linear term is kept, leading to

$$Q_{\text{lin}}(d_{\text{t}}|\bar{\boldsymbol{\mu}}, \mathbf{x}) = \bar{\boldsymbol{\mu}}'\boldsymbol{\Sigma}^{-1}\bar{\mathbf{f}} \ . \tag{4.23}$$

Realizing, that the channel compensated UBM is now a vector of zeros, and substituting (4.23) to (4.18), the formula for computing the LLR simplifies to

$$\text{LLR}_{\text{lin}}(d_{\text{t}}|\mathbf{s}, \mathbf{x}) = (\mathbf{V}\mathbf{y} + \mathbf{D}\mathbf{z})'\boldsymbol{\Sigma}^{-1}(\mathbf{f} - \mathbf{N}\mathbf{m} - \mathbf{N}\mathbf{c}) \ . \tag{4.24}$$

Since we are using UBM-aligned statistics, the log-likelihood ratios are lower-bound to the true log-likelihood ratio, as described in Section 3.3.1. Comparing the linear scoring and the LPT (quadratic) scoring, which both use the same UBM channel estimation, it is worth noting that the linear scoring can be a better approximation to the true LLR, as is shown in Figure 4.6.

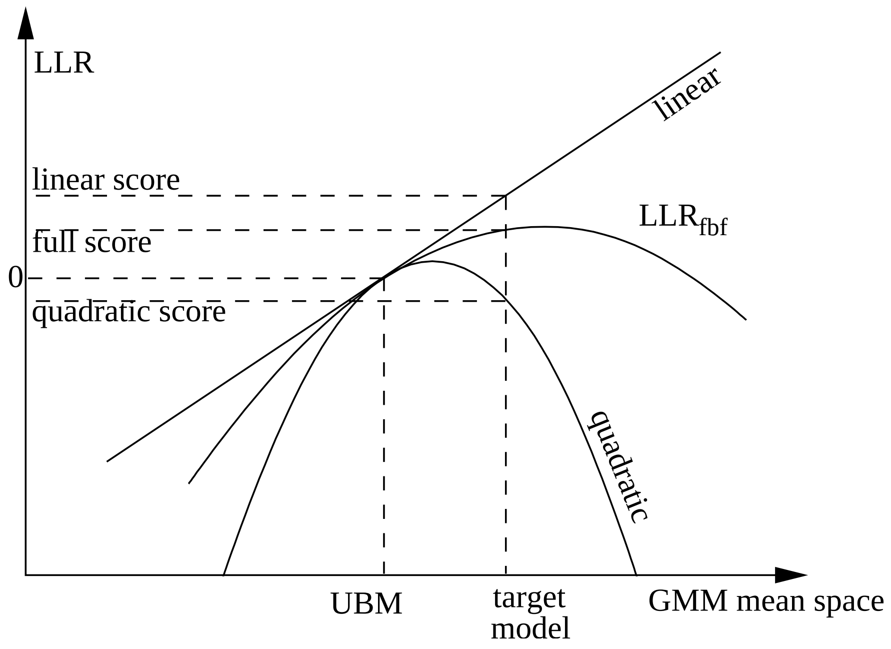


Figure 4.6: An illustration of the scoring behavior for frame-by-frame, quadratic (LPT), and linear scoring.

4.3 Experimental setup

4.3.1 Test Set

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

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.

4.3.2 Feature Extraction

In my experiments, I 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-frame window resulting in 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.

4.3.3 JFA Training

We used gender independent UBM containing 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 included 300 speaker factors, 100 channel factors, and diagonal matrix **D**. While **U** was trained on the NIST data, **D** and **V** were trained on two disjoint sets comprising NIST and Switchboard data.

4.3.4 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.

4.3.5 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 of 1024 KB. For the rest of the experiments, computing cluster was used.

4.4 Results

Table 4.1 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. 4.6 shows this case. Table 4.2 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. For the normalized case, however, all systems are very comparable. Let us note that the point estimate result is comparable to the full frame-by-frame scoring. The difference between the two approaches is only in the frame alignment, i.e. frame-by-frame uses the target model alignment, while point estimate uses the UBM fixed alignment, as described in Section 3.3.1. In that section, we have stated that the UBM fixed alignment is a good approximation to the target model alignment and this experiment confirms that assumption. Although mathematically very simple, the linear scoring gives comparable results to the other systems. Not only is this approach practical in terms of speed and implementation, but as we will see in Section 6.2, it is simple to derive through for the purpose of discriminative training.

Table 4.1: *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 4.2: *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

Table 4.3: *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**	$2.07e^{-3}$

4.4.1 Speed

The aim of this experiment was to show the approximate real time factor of each of the systems. As was mentioned in the introductory section, this experiment is targeted for laboratory usage. It does not compare the techniques in terms of absolute speed from the moment of acquiring the recording. Rather it lets the user extract the necessary information for faster experiments and system tweaking. 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 operations, such as reading of hyper-parameters, models, etc. were not comprised in the result. Each measuring was repeated 5 times and averaged. Table 4.3 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.

Chapter 5

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.

5.1 Theoretical background

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

$$\boldsymbol{\mu} = \mathbf{m} + \mathbf{T}\boldsymbol{\phi} \tag{5.1}$$

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 $\boldsymbol{\phi}$ is a standard normal distributed vector of size M. The matrix $\mathbf{T}$ is also sometimes referred to as "total variability subspace".

For each observation i, the aim is to estimate the parameters of the posterior probability of $\boldsymbol{\phi}$:

$$\mathrm{p}(\boldsymbol{\phi}|i) = \mathcal{N}(\boldsymbol{\phi}; \hat{\boldsymbol{\phi}}_i, \mathbf{L}_i^{-1}) \tag{5.2}$$

The i-vector $\boldsymbol{\phi}_i$ is the MAP point estimate of the variable $\boldsymbol{\phi}$, i.e. the mean $\hat{\boldsymbol{\phi}}_i$ of the posterior distribution $\mathrm{p}(\boldsymbol{\phi}|i)$.[1] It maps most of the relevant information from a variable-length observation i to a fixed- (small-) dimensional vector. $\mathbf{T}$ is referred to as the i-vector extractor. From this point of view, i-vectors can be understood as features for further recognition. The extraction of i-vectors as well as training the $\mathbf{T}$ matrix will be described in sections 5.1.2 and 5.1.3.

To apply the concept of GMM subspace modeling on i-vectors, as defined in Section 3.6, we can define the following mapping:

$$\begin{aligned} \underline{\mathbf{y}} &:= \boldsymbol{\phi} \\ \underline{\mathbf{V}} &:= \mathbf{T} \; . \\ \underline{\mathbf{m}} &:= \mathbf{m} \end{aligned} \tag{5.3}$$

Extracting an i-vector is then equal to computing the mean of the posterior distribution (5.2) using (3.42). Also, estimation of the hyper-parameters is performed using the algorithm as defined in Section 3.6.4. We will, however, rewrite the formula using trasformed statistics as defined in the following section.

5.1.1 Data

The input data for the observation i is given as a set of zero- and first-order statistics, as defined in Section 3.4. 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:

$$\begin{aligned} \mathbf{f}_i^{(c)} &\leftarrow \mathbf{f}_i^{(c)} - N_i^{(c)}\mathbf{m}^{(c)} \\ \mathbf{m}^{(c)} &\leftarrow \mathbf{0} \end{aligned}$$

This step technically leaves out the necessity of (3.43). 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:

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

where $\boldsymbol{\Sigma}^{(c)-\frac{1}{2}}$ is a symmetrical decomposition of an inverse of $\boldsymbol{\Sigma}^{(c)}$ (such as Cholesky decomposition), and $\mathbf{T}^{(c)}$ is an $F{\times}M$ sub-matrix of $\mathbf{T}$ corresponding to mixture component c such that

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}^{(1)} \\ \vdots \\ \mathbf{T}^{(C)} \end{bmatrix} . \tag{5.4}$$

The principle is illustrated in Figure 5.1. Note that we will not use any special symbols to denote this transformation. From this moment on, the statistics will be assumed to be presented in this form.

[1]In this work, we commonly refer to the means of distributions with a $\hat{\text{hat}}$ symbol, but since we use the i-vector as a feature, we omit the hat and use plain $\boldsymbol{\phi}$.

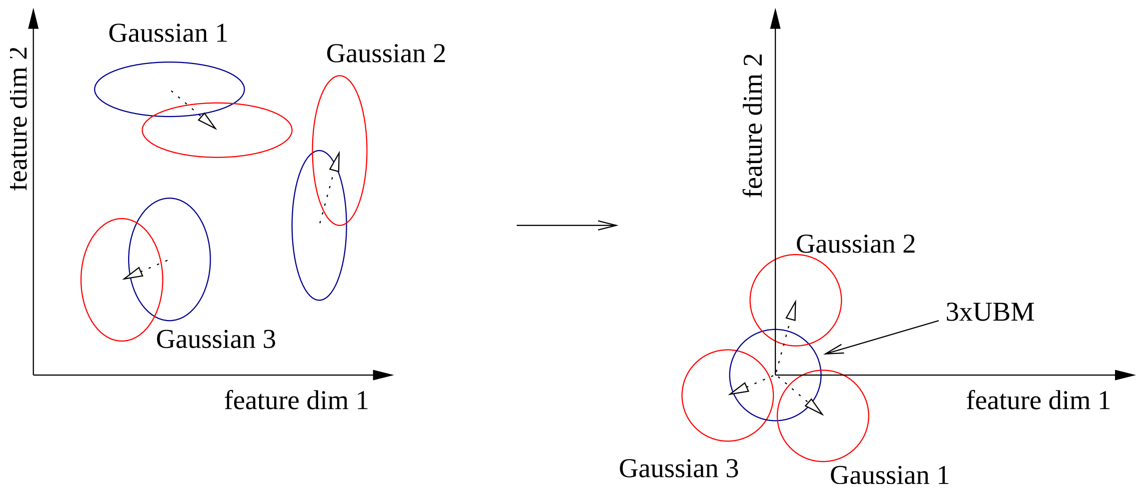


Figure 5.1: Illustration of data normalization—the blue ellipses represent the UBM covariances for different Gaussian components, while the red ellipses represent the adapted speaker model. The normalization "squashes" the covariances so that they become identities and are distributed around the center of the coordinate system.

5.1.2 i-vector Extraction Via Posterior Evaluation

Using the knowledge about GMM subspace modeling as described in Section 3.6, and using the symbol substitution (5.3), and with the data transforms from the previous section, the extraction of an i-vector for observation i is given as

$$\boldsymbol{\phi}_i = \hat{\boldsymbol{\phi}}_i = \mathbf{L}_i^{-1}\mathbf{T}'\mathbf{f}_i \;, \tag{5.5}$$

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

$$\mathbf{L}_i = \mathbf{I} + \sum_{c=1}^{C} N_i^{(c)}\mathbf{T}^{(c)\prime}\mathbf{T}^{(c)} \;. \tag{5.6}$$

5.1.3 Model Training

Model hyper-parameters $\mathbf{T}$ are estimated using the same EM and MD algorithm as shown in the GMM subspace modeling. Note that our algorithm performs the two steps jointly, i.e., we accumulate the statistics for the EM and MD at the same time (the E step) and then do a sequence of EM and MD update. We could also do the complete EM update first and the do the MD update, but this would take double the time due to the computational complexity of the posterior computation. In the E step, the following accumulators are collected using all training observations i:

$$\mathbf{C} = \sum_i \mathbf{f}_i \boldsymbol{\phi}_i' \tag{5.7}$$

$$\mathbf{A}^{(c)} = \sum_i N_i^{(c)} \left(\mathbf{L}_i^{-1} + \boldsymbol{\phi}_i \boldsymbol{\phi}_i'\right) \tag{5.8}$$

$$\bar{\mathbf{y}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\phi}_i \tag{5.9}$$

$$\mathbf{P}^{-1} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{L}_i + (\boldsymbol{\phi}_i - \bar{\mathbf{y}})(\boldsymbol{\phi}_i - \bar{\mathbf{y}})' \, , \tag{5.10}$$

where $\boldsymbol{\phi}_i$ and $\mathbf{L}_i$ are the estimates from (5.5) and (5.6) for observation i. The joint update is given as follows:

$$\mathbf{T}_{\text{em}}^{(c)} = \mathbf{C}\mathbf{A}^{(c)^{-1}} \tag{5.11}$$

$$\mathbf{T}_{\text{md}} = \mathbf{T}_{\text{em}} \mathbf{J} \, , \tag{5.12}$$

where $\mathbf{J}$ is a symmetrical decomposition of $\mathbf{P}^{-1}$, such that $\mathbf{J}\mathbf{J}' = \mathbf{P}^{-1}$ (e.g. Cholesky decomposition).

In Section 6.4, it will be further shown how the i-vector extractor can be trained discriminatively using the cross-entropy objective function.

5.2 Recognition using Cosine Distance

Once the i-vectors are extracted, recognition is performed in the symmetrical fashion, as described in Section 1.2. Several approaches have been proposed, all inspired by standard pattern-recognition techniques. In [Dehak et al., 2010], the authors analyze support vector machine approach as well as cosine-distance scoring with different flavors of channel compensation. Let us now briefly describe the cosine distance approach as used in this work.

Cosine distance of two vectors is given as their inner product, normalized by their Euclidian lengths:

$$\text{score}\left(\boldsymbol{\phi}_{\text{target}}, \boldsymbol{\phi}_{\text{test}}\right) = \frac{\langle \boldsymbol{\phi}_{\text{target}}, \boldsymbol{\phi}_{\text{test}} \rangle}{\|\boldsymbol{\phi}_{\text{target}}\| \|\boldsymbol{\phi}_{\text{test}}\|} \, . \tag{5.13}$$

Since i-vectors contain both useful and nuisance information, mere cosine-distance does not work properly in this task. In [Dehak et al., 2010], it was shown that it is necessary to filter out the channel information using linear discriminant analysis (LDA)—represented by matrix $\mathbf{A}$—followed by within-class normalization (WCCN)—represented by matrix $\mathbf{R}$. The transformed i-vector is then given as

$$\hat{\boldsymbol{\phi}} = \mathbf{R}'\mathbf{A}'\boldsymbol{\phi} \, . \tag{5.14}$$

Let us now describe the computation of the transformation matrices.

5.2.1 LDA

Linear discriminant analysis is a technique for multi-class separability analysis and dimensionality reduction. It is similar to PCA in that it analyzes the directions in a vector space with respect to the highest useful variability, however, PCA does not take into account any difference in class of the data.

Mathematically, given a set of D-dimensional samples $\mathbf{x}_i$ belonging to K classes $\mathcal{C}_k$, we look for a linear transformation

$$\mathbf{y}_i = \mathbf{A}'\mathbf{x}_i \ , \tag{5.15}$$

which transforms the input into D'-dimensional samples, where $\mathbf{A}$ is an $D \times D'$ transformation matrix, with $D' < K - 1$, which maximizes the Fisher criterion [Bishop, 2006]:

$$J(\mathbf{A}) = \operatorname{tr}\left\{(\mathbf{A}\boldsymbol{\Sigma}_{\text{wc}}\mathbf{A}')^{-1}(\mathbf{A}\boldsymbol{\Sigma}_{\text{ac}}\mathbf{A}')\right\} \ . \tag{5.16}$$

The $\boldsymbol{\Sigma}_{\text{wc}}$ and $\boldsymbol{\Sigma}_{\text{ac}}$ are within-class and across-class covariance matrices, respectively, defined as

$$\boldsymbol{\Sigma}_{\text{wc}} = \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)' \tag{5.17}$$

$$\boldsymbol{\Sigma}_{\text{ac}} = \sum_{k=1}^{K} N_k(\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})' \ , \tag{5.18}$$

with

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{x}_n \tag{5.19}$$

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{k=1}^{K} N_k\boldsymbol{\mu}_k \ . \tag{5.20}$$

The solution for $\mathbf{A}$ is then given by the eigen-vectors of $\boldsymbol{\Sigma}_{\text{wc}}^{-1}\boldsymbol{\Sigma}_{\text{ac}}$ which correspond to the D' largest eigen-values.

5.2.2 Within-Class Normalization

Within-class normalization was originally proposed to minimize a particular upper bound on the error rate [Hatch et al., 2006] in an SVM-based SRE system. It is a linear transformation $\mathbf{R}$, such that if applied to the i-vectors, the average within-class covariance matrix $\boldsymbol{\Sigma}_{\text{wc}}$, as defined by (5.17), becomes identity. The solution for $\mathbf{R}$ is then

$$\mathbf{R} = \operatorname{Chol}\left(\boldsymbol{\Sigma}_{\text{wc}}^{-1}\right) \ , \tag{5.21}$$

where $\operatorname{Chol}(\cdot)$ is a Cholesky decomposition function.

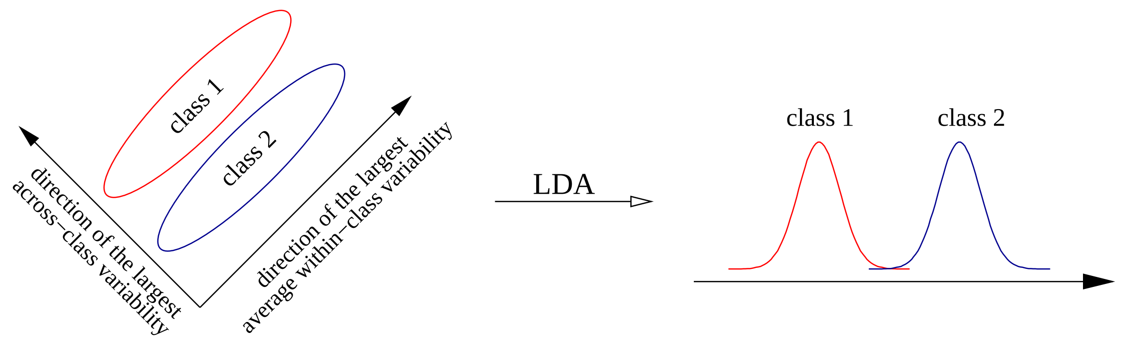

class 1
class 2
direction of the largest across–class variability
direction of the largest average within–class variability
LDA
class 1
class 2

Figure 5.2: Demonstration of LDA

5.3 PLDA

Another successful technique to facilitate comparison of i-vectors in a verification trial is through the Probabilistic Linear Discriminant Analysis (PLDA) model [Prince and Elder, 2007, Kenny, 2010]. Let us first assume a special kind of PLDA—the *two-covariance model*—in which the speaker variability and channel variability are assumed to be Gaussian and they are modeled using across-class and within-class variability matrices $\mathbf{\Sigma}_{\mathrm{ac}}$ and $\mathbf{\Sigma}_{\mathrm{wc}}$, respectively. Mathematically, the speaker identity is represented by a hidden variable $\mathbf{y}$ whose prior distribution is assumed to be

$$p(\mathbf{y}) = \mathcal{N}\left(\mathbf{y}; \boldsymbol{\mu}, \mathbf{\Sigma}_{\mathrm{ac}}\right) . \tag{5.22}$$

For a known speaker, represented by vector $\hat{\mathbf{y}}$, the distribution of i-vectors is given as

$$p(\boldsymbol{\phi}|\hat{\mathbf{y}}) = \mathcal{N}\left(\boldsymbol{\phi}; \hat{\mathbf{y}}, \mathbf{\Sigma}_{\mathrm{wc}}\right) . \tag{5.23}$$

Figure 5.3 depicts this situation.

In general PLDA, the covariance matrices do not necessarily have to be full-rank. Similarly to JFA, the i-vectors can then be decomposed as

$$\boldsymbol{\phi} = \boldsymbol{\mu} + \mathbf{V}\mathbf{y} + \mathbf{U}\mathbf{x} + \boldsymbol{\epsilon} , \tag{5.24}$$

where $\mathbf{V}$ describes the speaker subspace, $\mathbf{y}$ is a hidden variable representing the speaker, $\mathbf{U}$ describes the channel subspace, $\mathbf{x}$ is a hidden variable representing the channel, and $\boldsymbol{\epsilon}$ is a variable representing the residual data noise—note that it is not hidden as it can be computed once $\mathbf{x}$ and $\mathbf{y}$ are known. In its simplest form, PLDA imposes Gaussian priors on the variables:

$$\begin{aligned} p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I}) \\ p(\mathbf{x}) &= \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{I}) \\ p(\boldsymbol{\epsilon}) &= \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{D}^{-1}) , \end{aligned} \tag{5.25}$$

where $\mathbf{D}$ is a diagonal precision matrix of the residual data variability. The PLDA across-class covariance would then be given as $\mathbf{\Sigma}_{ac} = \mathbf{V}\mathbf{V}'$, and the within-class covariance would be $\mathbf{\Sigma}_{wc} = \mathbf{U}\mathbf{U}'$. In our experiments, we assume the mean $\boldsymbol{\mu}$ to be zero.

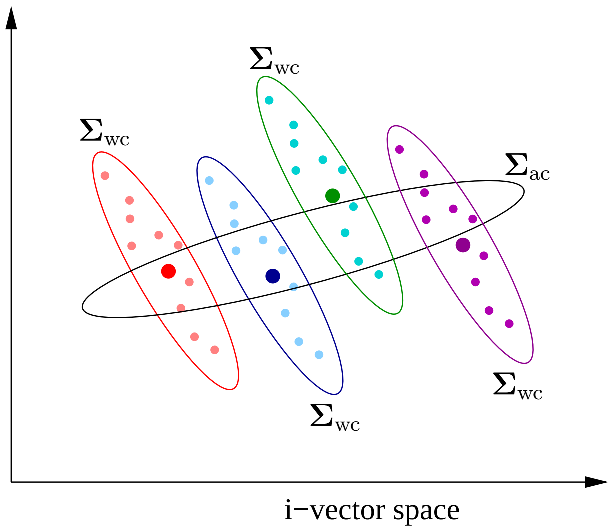


Figure 5.3: Demonstration of PLDA: the bold points represent the speaker identities in the i-vector space. Provided that we know the speaker identity $\mathbf{y}$, the conditional distribution of the i-vectors is given by the within-class covariances, depicted by the ellipses around the speaker identities.

Training such model in the ML fashion can be performed using the EM algorithm. The training procedure is described in Section A.3. Later in Section 6.3, we will again present how the parameters of PLDA can be optimized discriminatively.

A more complex model for PLDA was proposed in [Kenny, 2010], where Student's t-distribution was imposed on the prior. This "heavy-tail" (thus HTPLDA) model gives better accuracy, however there is no close-form solution for the posteriors, and both the training and testing are very complex and time-expensive algorithms. It was later shown that length normalization [Garcia-Romero, 2011] allows the Gaussian PLDA to achieve results comparable to HTPLDA. The reason is that length normalization forces the i-vectors to lie on a unity sphere, which brings them closer to the Gaussian distribution shell where most of the probability mass is concentrated. The transformation is given as

$$\bar{\boldsymbol{\phi}} = \frac{\boldsymbol{\phi}}{\|\boldsymbol{\phi}\|} = \frac{\boldsymbol{\phi}}{\sqrt{\boldsymbol{\phi}'\boldsymbol{\phi}}} \,. \tag{5.26}$$

5.3.1 Trial Scoring

Let us now remind that the speaker verification score is a function of a trial—i.e. a pair of i-vectors $\boldsymbol{\phi}_1$, $\boldsymbol{\phi}_2$—and that it is computed in the symmetrical way, i.e. we test whether the pair of i-vectors was generated by the same speaker ($\mathcal{H}_1$) or not ($\mathcal{H}_2$). The trial score is then defined as a log-likelihood ratio between the two hypotheses, as defined by (1.3), i.e.

$$s(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) = \log \frac{p(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2|\mathcal{H}_1)}{p(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2|\mathcal{H}_2)} \,. \tag{5.27}$$

Let us now consider the process of generating the two i-vectors under the different hypotheses:

- $\mathcal{H}_1$ In the case of a same-speaker trial, a single speaker factor $\hat{\mathbf{y}}$ is generated from the prior $p(\mathbf{y})$. The two trial i-vectors $\boldsymbol{\phi}_1$, $\boldsymbol{\phi}_2$ are then generated from $p(\boldsymbol{\phi}|\hat{\mathbf{y}})$. The joint likelihood for the two (independent) i-vectors being generated by the particular speaker $\hat{\mathbf{y}}$ is simply the product of the two likelihoods:

$$p(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2|\hat{\mathbf{y}}) = p(\boldsymbol{\phi}_1|\hat{\mathbf{y}})p(\boldsymbol{\phi}_2|\hat{\mathbf{y}}) \ . \tag{5.28}$$

  The likelihood that the two i-vectors are generated by any speaker (but still common to both i-vectors) is then computed by marginalization over all possible speakers:

$$\begin{aligned} p(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2|\mathcal{H}_1) &= \int p(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2|\mathbf{y})p(|\mathbf{y})\,\mathrm{d}\mathbf{y} && (5.29)\\ &= \int p(\boldsymbol{\phi}_1|\mathbf{y})p(\boldsymbol{\phi}_2|\mathbf{y})p(\mathbf{y})\,\mathrm{d}\mathbf{y} \ . && (5.30)\end{aligned}$$

- $\mathcal{H}_2$ In the case of a different-speaker trial, two different speaker factors $\hat{\mathbf{y}}_1$ and $\hat{\mathbf{y}}_2$ are generated from $p(\mathbf{y})$. The two trial i-vectors $\boldsymbol{\phi}_1$, $\boldsymbol{\phi}_2$ are then generated from $p(\boldsymbol{\phi}|\hat{\mathbf{y}}_1)$ and $p(\boldsymbol{\phi}|\hat{\mathbf{y}}_2)$, respectively. The joint likelihood for the two i-vectors being generated by the two speakers is simply the product of the two likelihoods:

$$p(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2|\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2) = p(\boldsymbol{\phi}_1|\hat{\mathbf{y}}_1)p(\boldsymbol{\phi}_2|\hat{\mathbf{y}}_2) \ . \tag{5.31}$$

  The marginal likelihood that the two i-vectors are generated by any two speakers is then computed as

$$\begin{aligned} p(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2|\mathcal{H}_2) &= \iint p(\boldsymbol{\phi}_1|\mathbf{y}_1)p(\boldsymbol{\phi}_2|\mathbf{y}_2)p(\mathbf{y}_1)p(\mathbf{y}_2)\,\mathrm{d}\mathbf{y}_1\,\mathrm{d}\mathbf{y}_2 && (5.32)\\ &= p(\boldsymbol{\phi}_1)p(\boldsymbol{\phi}_2) \ . && (5.33)\end{aligned}$$

Plugging the conditional likelihoods (5.30) and (5.33) into the log-likelihood ratio (5.27), we get

$$\begin{aligned} s(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) &= \log \frac{p(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2|\mathcal{H}_1)}{p(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2|\mathcal{H}_2)}\\ &= \frac{\int p(\boldsymbol{\phi}_1|\mathbf{y})p(\boldsymbol{\phi}_2|\mathbf{y})p(\mathbf{y})\,\mathrm{d}\mathbf{y}}{p(\boldsymbol{\phi}_1)p(\boldsymbol{\phi}_2)} \ . \end{aligned} \tag{5.34}$$

The integrals—which can be interpreted as convolutions of Gaussians—can be evaluated analytically, giving

$$\begin{aligned} s \ &= \ \log \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\phi}_1\\ \boldsymbol{\phi}_2\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\mu}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{tot} & \boldsymbol{\Sigma}_{ac}\\ \boldsymbol{\Sigma}_{ac} & \boldsymbol{\Sigma}_{tot}\end{bmatrix}\right)\\ &- \ \log \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\phi}_1\\ \boldsymbol{\phi}_2\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\mu}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{tot} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Sigma}_{tot}\end{bmatrix}\right) \ , \end{aligned} \tag{5.35}$$

where the total covariance matrix is given as $\mathbf{\Sigma}_{tot} = \mathbf{\Sigma}_{wc} + \mathbf{\Sigma}_{ac}$. By expanding the Gaussian computation and simplifying the formulas, we obtain a bi-linear form of the score:

$$\begin{aligned} s(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) = {} & \boldsymbol{\phi}_1^T \mathbf{\Lambda} \boldsymbol{\phi}_2 + \boldsymbol{\phi}_2^T \mathbf{\Lambda} \boldsymbol{\phi}_1 + \boldsymbol{\phi}_1^T \mathbf{\Gamma} \boldsymbol{\phi}_1 + \boldsymbol{\phi}_2^T \mathbf{\Gamma} \boldsymbol{\phi}_2 \\ & + (\boldsymbol{\phi}_1 + \boldsymbol{\phi}_2)^T \mathbf{c} + k \ , \end{aligned} \tag{5.36}$$

where

$$\mathbf{\Gamma} = -\frac{1}{4} (\mathbf{\Sigma}_{wc} + 2\mathbf{\Sigma}_{ac})^{-1} - \frac{1}{4}\mathbf{\Sigma}_{wc}^{-1} + \frac{1}{2}\mathbf{\Sigma}_{tot}^{-1} \tag{5.37}$$

$$\mathbf{\Lambda} = -\frac{1}{4} (\mathbf{\Sigma}_{wc} + 2\mathbf{\Sigma}_{ac})^{-1} + \frac{1}{4}\mathbf{\Sigma}_{wc}^{-1} \tag{5.38}$$

$$\mathbf{c} = \left((\mathbf{\Sigma}_{wc} + 2\mathbf{\Sigma}_{ac})^{-1} - \mathbf{\Sigma}_{tot}^{-1} \right) \boldsymbol{\mu} \tag{5.39}$$

$$\begin{aligned} k = {} & \log |\mathbf{\Sigma}_{tot}| - \frac{1}{2} \log |\mathbf{\Sigma}_{wc} + 2\mathbf{\Sigma}_{ac}| - \frac{1}{2} \log |\mathbf{\Sigma}_{wc}| \\ & + \boldsymbol{\mu}^T \left(\mathbf{\Sigma}_{tot}^{-1} - (\mathbf{\Sigma}_{wc} + 2\mathbf{\Sigma}_{ac})^{-1} \right) \boldsymbol{\mu} \ . \end{aligned} \tag{5.40}$$

5.4 Simplifications of i-vector extraction

Looking at the i-vector extraction in its basic form, as given by (5.5) and (5.6), we can now analyze its complexity in terms of computation power and memory. For convenience, let us repeat their form at this place.

$$\boldsymbol{\phi}_i = \mathbf{L}_i^{-1} \mathbf{T}' \mathbf{f}_i \tag{5.5}$$

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

$$\mathbf{L}_i = \mathbf{I} + \sum_{c=1}^{C} N_i^{(c)} \mathbf{T}^{(c)\prime} \mathbf{T}^{(c)} \ . \tag{5.6}$$

Let us repeat that C is number of Gaussian components, F is feature dimensionality, and M is the dimensionality of the i-vector. The computational complexity of the i-vector estimation for one observation is $O(CFM + CM^2 + M^3)$. The first term represents the $\mathbf{T}'\mathbf{f}_i$ multiplication. The second term represents the sum in (5.6) and includes the multiplication of $\mathbf{L}_i^{-1}$ with a vector. The third term represents the matrix inversion.

The memory complexity of the estimation is $O(CFM + CM^2)$. The first term represents the storage of all the input variables in (5.5), and the second term represents the pre-computed matrices in the sum of (5.6).

Note that the computation complexity grows quadratically with M in the sum of (5.6), and linearly with C. This becomes the bottle-neck in the i-vector computation, resulting in high memory and CPU demands.

For reference let us also recall that the update of the i-vector extractor parameters is given via the accumulators computed in the E step:

$$\mathbf{C} = \sum_i \mathbf{f}_i \boldsymbol{\phi}_i' \tag{5.7}$$

$$\mathbf{A}^{(c)} = \sum_i N_i^{(c)} \left(\mathbf{L}_i^{-1} + \boldsymbol{\phi}_i \boldsymbol{\phi}_i' \right) \tag{5.8}$$

$$\bar{\mathbf{y}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\phi}_i \tag{5.9}$$

$$\mathbf{P}^{-1} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{L}_i + (\boldsymbol{\phi}_i - \bar{\mathbf{y}})(\boldsymbol{\phi}_i - \bar{\mathbf{y}})' \,, \tag{5.10}$$

where $\boldsymbol{\phi}_i$ and $\mathbf{L}_i$ are the estimates from (5.5) and (5.6) for observation i. The joint update is given as follows:

$$\mathbf{T}_{\text{em}}^{(c)} = \mathbf{C}{\mathbf{A}^{(c)}}^{-1} \tag{5.11}$$

$$\mathbf{T}_{\text{md}} = \mathbf{T}_{\text{em}} \mathbf{J} \,, \tag{5.12}$$

where $\mathbf{J}$ is a symmetrical decomposition of $\mathbf{P}^{-1}$, such that $\mathbf{J}\mathbf{J}' = \mathbf{P}^{-1}$ (e.g. Cholesky decomposition).

5.4.1 Simplification 1: Approximating the Zero-order Statistics

Our main motivation for finding simplifications for i-vector extractor was speeding up the process of speaker verification in real-time systems and running robust speaker verification systems on small scale devices, such as mobile phones, and speeding up our experimental systems leading to faster turnover of experiments.

The most demanding step in i-vector extraction is the computation of the precision matrix $\mathbf{L}$, as defined by (5.6). In the first simplification, we apply the assumption that the GMM alignment for the zero-order statistics in (5.6) is constant for all frames, i.e. the posterior occupation probabilities $\gamma^{(c)}$ are replaced by their prior probabilities represented by the UBM GMM weights $w^{(c)}$. The new zero-order statistics are then given by scaling the GMM weights by the total number of frames in an utterance:

$$\tilde{N}_i^{(c)} = w^{(c)} N_i \,, \tag{5.41}$$

where $w^{(c)}$ is the GMM UBM weight of component c, and N_i is the number of frames in an utterance i. Substituting $N_i^{(c)}$ in (5.6) by $\tilde{N}_i^{(c)}$ from (5.41), we get

$$\tilde{\mathbf{L}}_i = \mathbf{I} + N_i \mathbf{W} \,, \tag{5.42}$$

where

$$\mathbf{W} = \sum_{c=1}^{C} w^{(c)} \mathbf{T}^{(c)\prime} \mathbf{T}^{(c)} \,. \tag{5.43}$$

We can exploit this simplification in the i-vector extractor training procedure as well by using the new posterior approximation. First of all, $\boldsymbol{\phi}_i$ in (5.7) through (5.9) would be estimated using the approximated $\tilde{\mathbf{L}}_i$. Second, $\mathbf{L}_i$ in (5.8) would be substituted for the approximated $\tilde{\mathbf{L}}_i$.

Note that $\mathbf{W}$ in (5.43) 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 complexity of this algorithm reduces to $O(CFM + M^3)$ with the dominating inversion step. The memory complexity reduces to $O(CFM + M^2)$.

5.4.2 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 $\mathbf{L}_i$, which would need to be rotated back using $\mathbf{G}$. We can then express (5.6) as

$$\mathbf{L}_i = \mathbf{G}^{(-1)'}\tilde{\mathbf{L}}_i\mathbf{G}^{-1} , \tag{5.44}$$

where

$$\tilde{\mathbf{L}}_i = \mathbf{G}'\mathbf{G} + \sum_{c=1}^{C} N_i^{(c)}\mathbf{G}'\mathbf{T}^{(c)'}\mathbf{T}^{(c)}\mathbf{G} . \tag{5.45}$$

Assuming that $\tilde{\mathbf{L}}_i$ is diagonal, we can rewrite it as

$$\tilde{\mathbf{L}}_i = \mathrm{Diag}\left(\mathrm{diag}(\mathbf{G}'\mathbf{G}) + \mathbf{V}\mathbf{n}_i\right) , \tag{5.46}$$

where $\mathbf{V}$ is a $M{\times}C$ matrix whose c-th column is $\mathrm{diag}(\mathbf{G}'\mathbf{T}^{(c)'}\mathbf{T}^{(c)}\mathbf{G})$. $\mathrm{Diag}(\cdot)$ maps a vector to a diagonal matrix, while $\mathrm{diag}(\cdot)$ maps a matrix diagonal to a vector. Combining (5.44) and (5.5), we get

$$\tilde{\boldsymbol{\phi}}_i = \mathbf{G}\tilde{\mathbf{L}}_i^{-1}\mathbf{G}'\mathbf{T}'\mathbf{f}_i . \tag{5.47}$$

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 multiplication by $\mathbf{V}$ in (5.46).

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

Eigen-decomposition

Let $\mathbf{W}$ be the weighted average of $\mathbf{T}^{(c)'}\mathbf{T}^{(c)}$ as defined by (5.43). We assume $\mathbf{W}$ to be a full-rank matrix with M linearly independent eigenvectors. Then $\mathbf{W}$ can be factorized as

$$\mathbf{W} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{-1} . \tag{5.48}$$

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 $\mathbf{G} = \mathbf{Q}$.

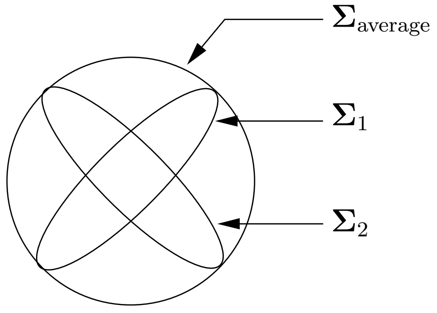


Figure 5.4: Example of a two-class problem, where both PCA and LDA would fail

Heteroscedastic Linear Discriminant Analysis

If the average matrix $\mathbf{W}$ from (5.43) is close to diagonal, then the eigen-decomposition does not have to be effective in diagonalizing the per-component quadratic terms $\mathbf{T}^{(c)\prime}\mathbf{T}^{(c)}$. Le us reformulate the problem by considering the quadratic terms as being pseudo-covariance matrices. An extreme example is shown in Figure 5.4. We see that the distributions given by the covariance matrices are perpendicular to each other. When they get averaged to compute the PCA transform, the new covariance is clearly diagonal and no PCA is needed.

The Heteroscedastic Linear Discriminant Analysis (HLDA) can be used to derive linear projection de-correlating the space. It is a supervised method, which allows us to derive such projection that best de-correlates features associated with each particular class (maximum likelihood linear transformation for diagonal covariance modeling [Kumar, 1997]). The features are described by the corresponding within-class covariance matrices. In our task, the classes are defined as GMM components and we use the within-class pseudo-covariance matrices $\mathbf{W}_{\text{wc}}^{(c)}$, given as

$$\mathbf{W}_{\text{wc}}^{(c)} = \mathbf{T}^{(c)\prime}\mathbf{T}^{(c)} . \tag{5.49}$$

Note again that $\mathbf{W}_{\text{wc}}^{(c)}$ are not real covariance matrices (therore the term "pseudo-covariance"), but our problem of simultaneously orthogonalizing many quadratic terms $\mathbf{W}_{\text{wc}}^{(c)}$ is very similar to that solved by HLDA. Therefore we apply the same algorith to deal with our problem. An efficient iterative algorithm [Gales, 1999b] was used in our experiments to estimate matrix $\hat{\mathbf{G}}$ from previous estimate $\mathbf{G}$. Each new k-th column $\hat{\mathbf{g}}_k$ of $\hat{\mathbf{G}}$ is given as

$$\hat{\mathbf{g}}_k = \mathbf{c}_k{}'\mathbf{H}^{(k)^{-1}}\sqrt{\frac{1}{\mathbf{c}_k{}'\mathbf{H}^{(k)^{-1}}\mathbf{c}_k}} \tag{5.50}$$

where $\mathbf{c}_k{}'$ is the k-th row vector of co-factor matrix

$$\mathbf{C} = |\mathbf{G}|\mathbf{G}^{-1} , \tag{5.51}$$

and

$$\mathbf{H}^{(k)} = \sum_{c=1}^{C} \frac{w^{(c)}}{\mathbf{g}_k{}'\mathbf{W}_{\text{wc}}^{(c)}\mathbf{g}_k}\mathbf{W}_{\text{wc}}^{(c)} , \tag{5.52}$$

where $\mathbf{g}_k$ is the k-th column of the current estimate $\mathbf{G}$, and $w^{(c)}$ is the weight of GMM component c.

Note that LDA can be seen as special case of HLDA, where the covariance matrices of all classes are assumed to be identical.

5.4.3 The NIST Experiments

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.

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.

Two (gender-dependent) i-vector extractors were trained on 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 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.

Scoring and Normalization

The extracted i-vectors were scaled down using an LDA matrix to 200 dimensions, and further normalized by a within-class covariance matrix (see Section 5.2, [Dehak et al., 2010]). 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 scoring function (see Section 5.2). The scores were normalized using gender-dependent s-norm with a cohort of 400 speakers having 2 utterances per speaker.

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)

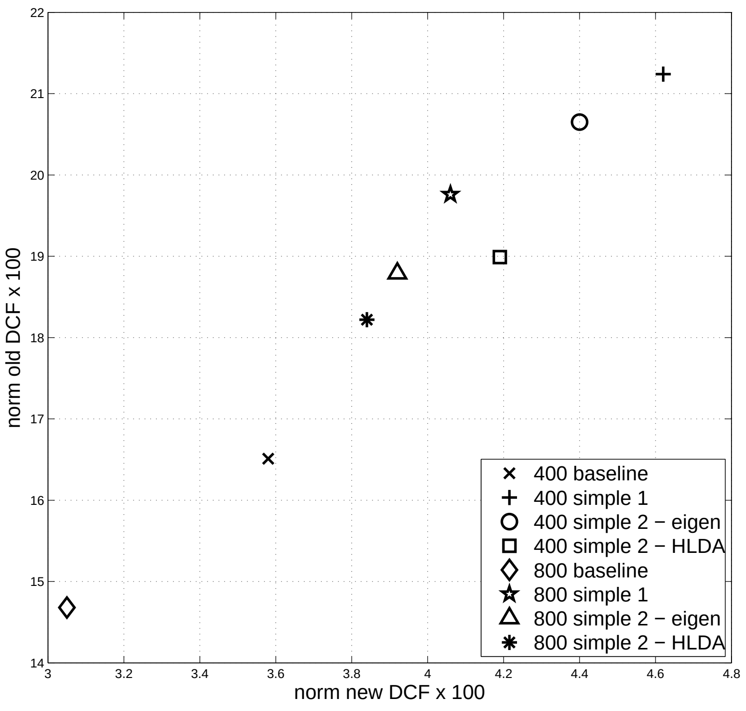


Figure 5.5: Constellation plot of the individual systems

dataset [NIST, nd]. The recognition accuracy is given as a set of equal error rate (EER), and the normalized DCF with parameters defined in the NIST 2010 SRE task ($\mathrm{DCF}_{\mathrm{new}}$) and in the previous SRE evaluations ($\mathrm{DCF}_{\mathrm{old}}$).

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

Table 5.1 summarizes the systems with respect to verification accuracy. Fig. 5.5 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”.

Speed and Memory

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 precomputed (thus not included in the reported times), and comprised in the algorithm-specific memory requirements:

- $\mathbf{T}^{(c)\prime}\mathbf{T}^{(c)}$ in (5.6)

Table 5.1: *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

Table 5.2: *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%

- $\mathbf{W}$ in (5.43)
- $\mathbf{G}$ and $\mathbf{T}^{(c)}\mathbf{G}$ in (5.44) and (5.47), and $\mathbf{V}$ in (5.46)

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.

As described earlier in Sec. 5.4.3, the computation time does not include reading of the necessary data and pre-computation of some terms. The results are reported in Tab. 5.2. The dominating complexity of matrix inversion makes “simple 2” faster than “simple 1”, as described in Sec. 5.4.1 and 5.4.2.

Tab. 5.4 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. 5.4.3). Projecting this linear transformation directly into the leftmost $\mathbf{G}$ of (5.47) could further decrease the complexity of the “simple 2” algorithm.

Table 5.3: *Time complexity for different systems. The first term is constant across different methods and represents* $\mathbf{V}'\mathbf{f}$ *for* N *observations*

method	time
baseline	$O(C^2F^2MN + CM^2N + M^3N)$
simple 1	$O(C^2F^2MN + M^3N)$
simple 2	$O(C^2F^2MN + CMN)$

Table 5.4: *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

5.4.4 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 5.5 shows the difference.

Time and memory complexity of collecting the accumulators $\mathbf{A}$ from (5.8) is almost identical to the computation of $\mathbf{L}_i$ in (5.6). The proposed method still keeps the same accumulator collection, however, avoiding the expensive computation of (5.6) decreases the E step time and memory complexity by a factor of 2.

Table 5.5: *Comparison of the proposed i-vector extractor training methods in terms of normalized DCFs and EER for NIST2010*

	DCF_{new}	DCF_{old}	EER
400 baseline	0.5460	0.1722	3.40
400 simple 1	0.5376	0.1729	3.42

Table 5.6: *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

5.4.5 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 (as described in Section 2.2.2.)

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.

As was mentioned in Section 2.4.8, 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. 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 5.6 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.

Chapter 6

Discriminative Training

Discriminative training has shown to be effective in numerous fields of speech processing. In LVCSR, acoustic models have been trained using different discriminative objectives such as Maximum Mutual Information (MMI), Minimum Classification Error (MCE), Minimum Phoneme Error (MPE), whose description and extensive study can be found in [Povey, 2004]. In Language Recognition, discriminative training has been successfully used in training the target language models (e.g. [Matějka, 2009, Campbell et al., 2006a]) or calibration (e.g. [Brümmer and van Leeuwen, 2006]). In Speaker Recognition, the use of discriminative training has been very limited. Let us do a short review of what has been done and give the motivation for this work. Let us start by looking at Figure 6.1 to remind the principles of the fully-generative modeling.

SVMs and their one-against-many strategy has been successfully used by different sites [Campbell et al., 2006b]. However, the objective of SVMs does not directly address the discrimination between the same- and different-speaker hypotheses. Rather, the SVM was trained so that it best separates the target-speaker data from the data belonging to a cohort of impostor speakers. This approach addresses only half of the SRE task. The full problem is to (i) distinguish one speaker from the others (as traditional SVM approach does) and (ii) to recognize the same speaker under different conditions, which the SVM does not address. Figure 6.2 shows the discriminative optimization in the task of SRE.

In fusion, Logistic Regression has been widely used (see [Brümmer and du Preez, 2006, Brümmer et al., 2007] and various system descriptions in [NIST, nd]). As opposed to SVM's, its objective is to directly maximize the overall discrimination of the system. Scores from two or more systems are linearly combined to maximize the system's cross-entropy as defined in Section 2.3.5. In this task, however, the discriminative model only combines systems, but it does not make them better. Figure 6.3 shows the situation.

In [Burget et al., 2011], we have shown that discriminatively training the PLDA parameters can lead to improvement in recognition performance. The objective is essentially the same as for fusion, however, it addresses optimizing the parameters of the system to make the system itself better. Figure 6.4 shows how this approach is different from the case of the SVMs. The discriminative optimization of the hyper-parameters is directly dependent on the score.

In [Burget et al., 2008], Lukáš Burget and Niko Brümmer used the same objective to discriminatively train the eigen-voices matrix in JFA. In that work, the scoring function

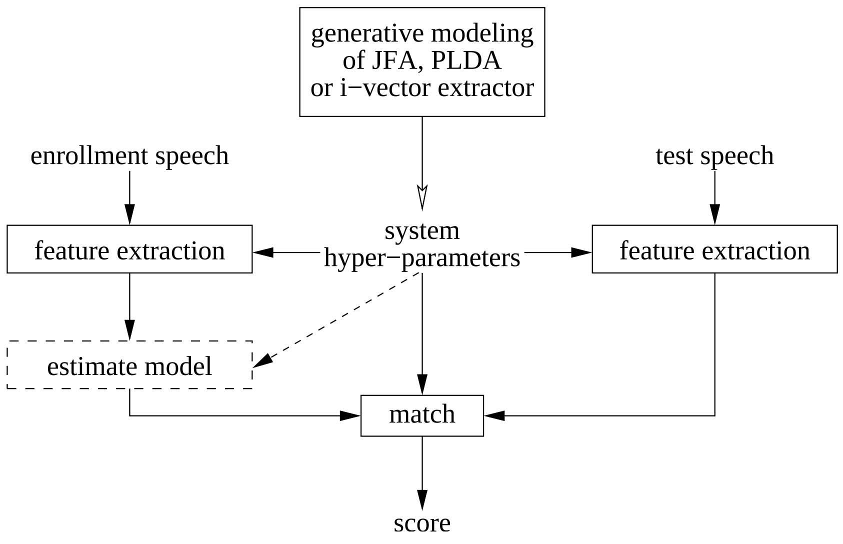


Figure 6.1: Fully generative approach—the dotted part is omitted for symmetrical systems.

for the discriminative objective was given by the linear scoring, as defined by (4.24).

In [Glembek et al., 2011a] we have shown that similar discriminative training framework can be adopted for training the parameters of the i-vector extractor. The work follows on the JFA discriminative training in that it addresses the optimization of the GMM subspace, however, the system setup makes the optimization more complex as the parameters lie "deeper" in the speaker recognition chain.

In this work, we review the techniques of discriminatively training JFA and PLDA, including the results, and we follow up the work by studying discriminative training of the i-vector extractor.

6.1 Discriminative Objective Function

Discriminative training is generally based on optimizing the parameters of a model with the objective of best class discrimination. In this work, our objective is to maximize the *posterior probabilities* of the classes given a set of training samples. Unlike generative training, where the models are trained as if they were to literally generate the training data for the individual classes, discriminative training aims at finding such model that best discriminates among the classes. In other words, discriminative objective is to classify training data correctly. The classification score of each trial is evaluated using some loss (or error) function, which is then summed over all training set to compute the overall objective. Having constructed the objective this way, the task of discriminative optimization is to minimize this overall loss (or to minimize the error).

Let x be a trial to be classified as either same-speaker (hypothesis $\mathcal{H}_1$) or different-speaker (hypothesis $\mathcal{H}_2$). Let $p(x|\mathcal{H}_j)$ be the likelihood of trial x given hypothesis $\mathcal{H}_j$.

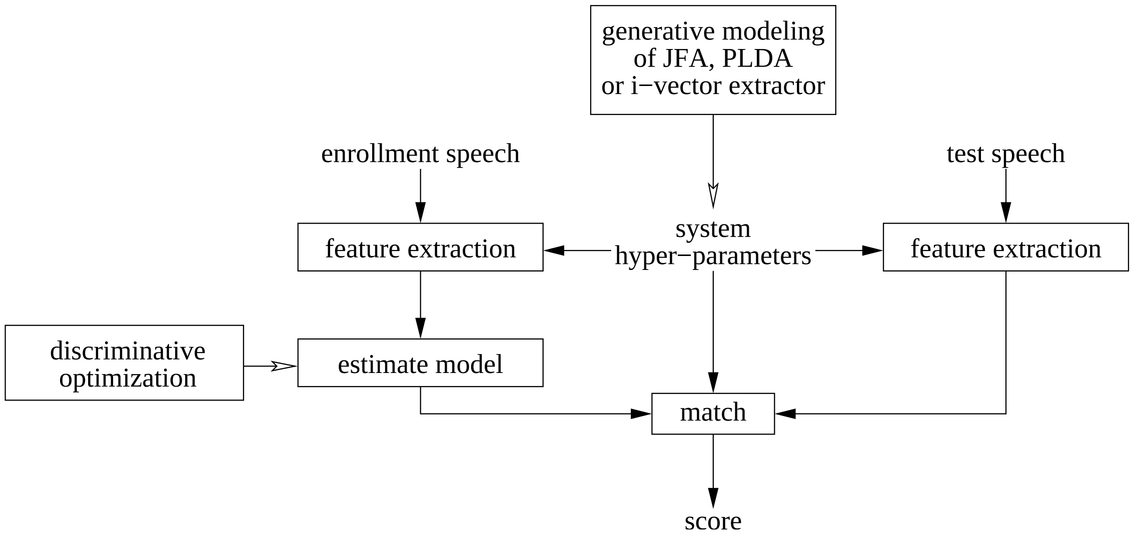


Figure 6.2: Discriminative optimization of the speaker model—note that we do not optimize the system parameters; rather the model is trained so that it best separates the target-speaker data from a set of non-target speakers.

The hypothesis posterior can then be evaluated using the Bayes rule:

$$p(\mathcal{H}_j|x) = \frac{p(x|\mathcal{H}_j)p(\mathcal{H}_j)}{p(x|\mathcal{H}_1)p(\mathcal{H}_1) + p(x|\mathcal{H}_2)p(\mathcal{H}_2)} \ . \tag{6.1}$$

We have expressed the hypothesis posterior in terms of likelihoods but we do not necessarily have those at hand. Instead, each trial is provided with a score s_x which we assume to be a log-likelihood ratio. From (6.1) it holds that

$$\frac{p(\mathcal{H}_1|x)}{p(\mathcal{H}_2|x)} = \frac{p(x|\mathcal{H}_1)p(\mathcal{H}_1)}{p(x|\mathcal{H}_2)p(\mathcal{H}_2)} \ , \tag{6.2}$$

and so

$$\begin{aligned}\log \frac{p(\mathcal{H}_1|x)}{p(\mathcal{H}_2|x)} &= \log \frac{p(x|\mathcal{H}_1)}{p(x|\mathcal{H}_2)} + \log \frac{p(\mathcal{H}_1)}{p(\mathcal{H}_2)} \\ &= s_x + \operatorname{logit} p(\mathcal{H}_1) \\ &= \hat{s}_x \ , \end{aligned} \tag{6.3}$$

where $\hat{s}$ denotes a *log-posterior-ratio*, and

$$\operatorname{logit} p(\mathcal{H}_1) = \log \frac{p(\mathcal{H}_1)}{p(\mathcal{H}_2)} \tag{6.4}$$

$$= \log \frac{p(\mathcal{H}_1)}{1 - p(\mathcal{H}_1)} \ . \tag{6.5}$$

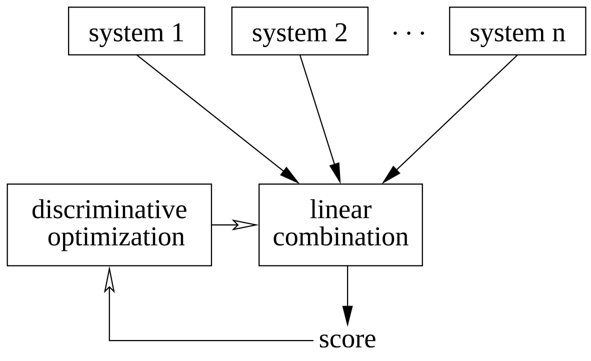


Figure 6.3: Discriminative optimization of the fusion—the fusion makes the overall system better by discriminatively optimizing the scale and shift parameters of the individual systems, but it does not make the systems better recognizers.

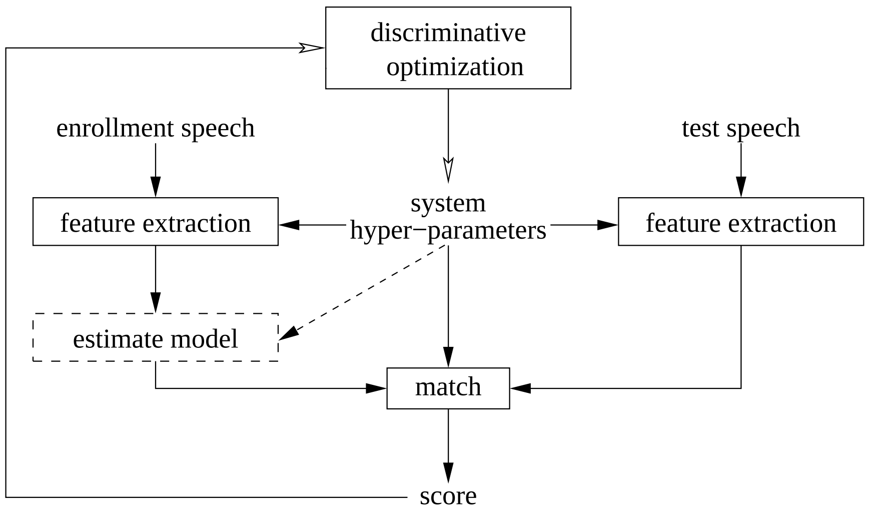


Figure 6.4: Discriminative optimization of the system hyper-parameters

We can rewrite the posteriors as

$$p(\mathcal{H}_1|x) = \sigma\left(-\hat{s}_x\right) \tag{6.6}$$

$$\begin{aligned} p(\mathcal{H}_2|x) &= 1 - p(\mathcal{H}_1|x) \\ &= \sigma\left(\hat{s}_x\right) \ , \end{aligned} \tag{6.7}$$

where $\sigma(\cdot)$ is the the logistic function defined as

$$\sigma(s) = \frac{1}{1 + e^{-s}} \ . \tag{6.8}$$

Note that we assume the hypotheses' priors to be equal, which sets $\hat{s}_x = s_x$ and we will simply use the log-likelihood-ratio instead of the log-posterior-ratio throughout the rest of the work[1].

[1]To change the class priors for the optimization, we can simply add the logit $p(\mathcal{H}_1)$ constant to the

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 x a log-likelihood ratio s and the correct label t, the log probability of recognizing the trial correctly can be expressed as

$$\log p\,(t|s) = -\log\left(1 + e^{-ts}\right) \;. \tag{6.9}$$

In the case of logistic regression, the objective function to be maximized is the log probability of correctly classifying all training examples with respect to the optimized parameters θ, i.e., the sum of expressions (6.9) evaluated for all training trials $\mathcal{X}$:

$$\begin{aligned} \mathcal{Q}(\theta) &= \sum_{x \in \mathcal{X}} \log p\,(t_x|s_x) && (6.10)\\ &= \sum_{x \in \mathcal{X}} -\log\left(1 + e^{-t_x s_x}\right) \;, && (6.11) \end{aligned}$$

where

$$s_x = s_x(\theta) \;. \tag{6.12}$$

Equivalently, this can be expressed by minimizing the cross-entropy error function, whose more general form is

$$E_{\mathcal{X}}(\theta) = \sum_{x \in \mathcal{X}} \alpha_x E_{LR}\left(t_x s_x(\theta)\right) + R(\theta) \;, \tag{6.13}$$

where the logistic regression loss function

$$E_{LR}(ts) = \log(1 + \exp(-ts)) \tag{6.14}$$

is simply the negative log probability (6.9) of correctly recognizing a trial, α_x allows to scale each trial's loss (to be discussed later), and $R(\theta)$ is a regularization term which penalizes the objective function in case the parameters do not follow the regularizer conditions. In this work, we will use the L_2 regularizer which can be seen as imposing an isotropic Gaussian prior on the parameters [Bishop, 2006]:

$$R(\theta) = \frac{\lambda}{2}\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|^2 \;, \tag{6.15}$$

where bold-face $\boldsymbol{\theta}$ denotes a vector of stacked parameters θ, λ is a constant controlling the trade-off between the error function and the regularizer, i.e., the variance of the isotropic Gaussian prior, and $\hat{\boldsymbol{\theta}}$ defines the mean of the prior. Note that this kind of regularization is similar to the sum-of-squares penalty; however, it extends it by controlling the distance from some offset. The coefficients α_x allow us to weight individual trials, e.g. to balance the dataset, exclude suspicious trials, etc. In particular, they are useful to compensate for different numbers of the same-speaker and different-speaker trials. Note that when

scores.

setting them to $\frac{1}{2\log 2}\frac{1}{|\mathcal{X}_1|}$ and $\frac{1}{2\log 2}\frac{1}{|\mathcal{X}_2|}$, where $|\mathcal{X}_1|$ and $|\mathcal{X}_2|$ are the numbers of same- and different-speaker trials, respectively, and when λ is set to 0, we get

$$\begin{aligned} E_{\mathcal{X}}(\theta) &= \sum_{x\in\mathcal{X}_1} \frac{1}{2\log 2}\frac{1}{|\mathcal{X}_1|}\log\left(1+e^{-s_x(\theta))}\right) \\ &\quad + \sum_{x\in\mathcal{X}_2} \frac{1}{2\log 2}\frac{1}{|\mathcal{X}_2|}\log\left(1+e^{s_x(\theta)}\right) \qquad (6.16) \\ &= \frac{1}{2\log 2}\left(\frac{1}{|\mathcal{X}_1|}\sum_{x\in\mathcal{X}_1}\log\left(1+e^{-s_x(\theta)}\right) + \frac{1}{|\mathcal{X}_2|}\sum_{x\in\mathcal{X}_2}\log\left(1+e^{s_x(\theta)}\right)\right) \qquad (6.17) \\ &= C_{\mathrm{llr},\theta}(\mathcal{X}) \;, \qquad (6.18) \end{aligned}$$

which is the $C_{\mathrm{llr}}(\mathcal{X})$ objective (with respect to the parameters θ) for a set $\mathcal{X}$ as defined by (2.7) early in Section 2.3.5.

SVM Loss function

For completeness, let us realize that the E_{LR} as defined by (6.14) is not the only loss function that we can use. Taking (6.13) and replacing $E_{LR}(ts)$ with hinge loss function

$$E_{SV}(ts) = \max(0, 1-ts), \qquad (6.19)$$

we obtain an objective function typically used in SVM. SVM is a classifier traditionally understood to maximize the margin separating the class samples. Alternatively, one can see the hinge loss function as a piecewise approximation to the logistic regression loss function. Figure 6.5 shows the plot of these functions. We mention this loss function to realize the context and similarity of the two classifiers. Also, later in the experimental section of PLDA, parallel work to the one described was done by Sandro Cumani [Cumani, 2012] who used SVMs extensively in his work and the results are presented for comparison.

6.1.1 Gradient Evaluation

In order to optimize the parameters θ, we need to evaluate the gradient of the error function. Either we set it to zero and find the solution analytically, or—as used in this work—we use it to numerically optimize the parameters. The problem is that we are going to optimize parameters that are deep in the speaker recognition chain, i.e., the objective function is given as a composition of several functions. The solution is to find partial derivatives (Jacobian matrices) and to use the chain rule to compose the final gradient. In its general form, the gradient of the cross entropy objective is given as

$$\nabla E(\theta) = \sum_{x\in\mathcal{X}} \alpha_x \frac{\partial E(t_x s_x)}{\partial s_x}\frac{\partial s_x}{\partial \boldsymbol{\theta}} + \lambda(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}), \qquad (6.20)$$

where the derivation of the loss function $E(t_x s_x)$, w.r.t. score s_x, depends on the particular choice of the loss function. For the logistic regression loss function, it is defined as

$$\frac{\partial E_{LR}(ts)}{\partial s} = -t\sigma(-ts) \;, \qquad (6.21)$$

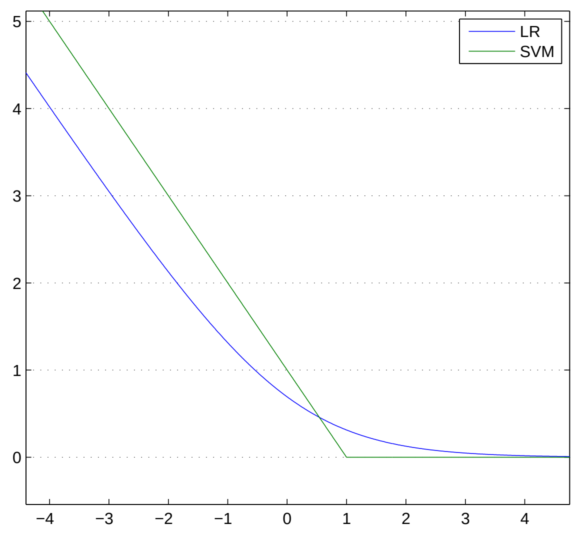


Figure 6.5: Loss functions for discriminative optimization: the logistic regression in blue and SVM hinge loss in green. Note the similarity of the functions.

where $\sigma(\cdot)$ is the logistic function as defined by (6.8). For the hinge loss function, the derivative is

$$\frac{\partial E_{SV}(ts)}{\partial s} = \begin{cases} 0 & \text{if} \quad ts \geq 1 \\ -t & \text{otherwise.} \end{cases} \tag{6.22}$$

To apply the discriminative optimization for a particular problem, one needs to define $\frac{\partial s_x}{\partial \boldsymbol{\theta}}$, which will be studied separately in the following sections.

6.2 Discriminative Training of JFA

This section reviews the work of [Burget et al., 2008], specifically the experiments of discriminatively training the JFA eigenvoice matrix $\mathbf{V}$. The task was to minimize the discriminative objective function as given by equation (6.16). The score $\bar{s}_x$ for the objective was given as

$$\bar{s}_x = \alpha s_x + \beta \ , \tag{6.23}$$

where s_x is the raw log-likelihood-ratio obtained from the full JFA model and α and β are parameters for a linear calibration of the scores to ensure that the objective function gets proper log-likelihood ratio. Since the simple linear scoring described in section 4.2.5 provided superior performance, it was the natural choice for these experiments. Therefore, the log-likelihood-ratio defined by (4.24) is used as the raw score s_x, generally leading to

$$\bar{s}_x = \alpha(\mathbf{V}\mathbf{y}_x + \mathbf{D}\mathbf{z}_x)'\boldsymbol{\Sigma}^{-1}\bar{\mathbf{f}}_x + \beta \ . \tag{6.24}$$

where

$$\bar{\mathbf{f}}_x = \mathbf{f}_x - \mathbf{N}_x(\mathbf{m} - \mathbf{U}\mathbf{x}_x) \ . \tag{6.25}$$

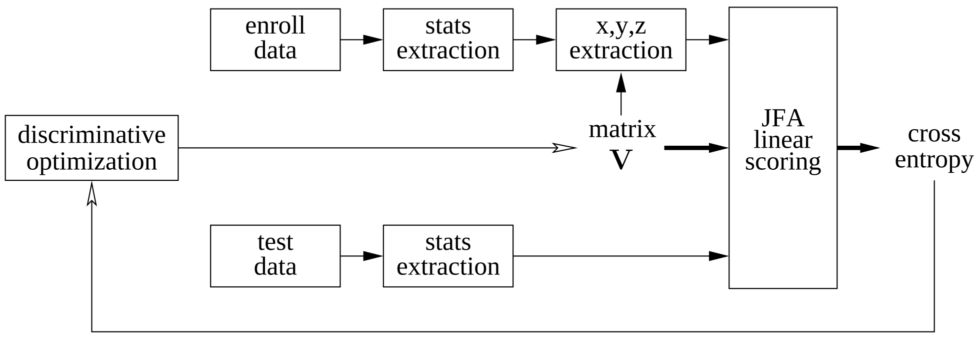


Figure 6.6: System setup for the discriminative training of the eigen-voices matrix $\mathbf{V}$. Note that when optimizing the parameters, we do not derive through the extraction of $\mathbf{V}$, rather we derive through linear scoring, which is depicted by the bold arrows.

Another advantage of using the linear scoring is that derivatives of the objective function w.r.t. $\mathbf{V}$ are easy to derive and can be very efficiently calculated. In all experiments described in this section, speaker factors for all training and enrollment utterances were precomputed using ML-trained JFA system and stayed fixed.

The full parameter set for optimization is $\theta = \langle \alpha, \beta, \mathbf{V} \rangle$. The optimization is an iterative process, where, in each iteration, we first fully optimize parameters α and β using linear logistic regression[2], which is followed by one iteration of modified gradient descent update of eigenvoices $\mathbf{V}$.

The gradient of the objective function has been partially derived in (6.20). We use the setup of α_x and λ as defined for the C_{llr} function in (6.17). Now we derive the score function (6.24) w.r.t. the parameters $\mathbf{V}$, and we obtain

$$\frac{\partial \bar{s}_x}{\partial \mathbf{V}} = \alpha \boldsymbol{\Sigma}^{-1} \bar{\mathbf{f}} \mathbf{y}' \, . \tag{6.26}$$

The complete gradient of the objective function w.r.t. the parameters is then constructed by plugging (6.26) into (6.20):

$$\nabla E(\mathbf{V}) = -\alpha \sum_{x \in \mathcal{X}} \alpha_x t_x \sigma(-t_x \bar{s}_x) \boldsymbol{\Sigma}^{-1} \bar{\mathbf{f}}_x \mathbf{y}_x{}' \, . \tag{6.27}$$

To optimize our objective function, we need to define a set of training trials $\mathcal{X}$ (composed of same- and different-speaker trials $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively). In these experiments, each possible pair of two segments from our training set formed a valid trial, where one segment is considered to be the enrollment and the other the test segment. This allows us to define $X \times X$ matrix $\mathbf{P}$, where $X = |\mathcal{X}|$ is number of the segments in the training set $\mathcal{X}$ and each element of the matrix corresponds to one trial where row index defines test segment and column index defines the enrollment segment. Let the elements of the matrix $\mathbf{P}$ corresponding to trial x be $\frac{1-p(\mathcal{H}_1|x)}{|\mathcal{X}_1|}$ if the trial is a target trial and $-\frac{p(\mathcal{H}_1|x)}{|\mathcal{X}_2|}$ if the trial is

[2]FoCal toolkit (http://niko.brummer.googlepages.com/focal) by Niko Brümmer was used for this purpose.

a non-target trial. Taking (6.27) and making use of matrix **P**, we obtain

$$\nabla E(\mathbf{V}) = \frac{1}{2\log 2}\alpha \boldsymbol{\Sigma}^{-1}\bar{\mathbf{F}}_{\mathcal{X}}\mathbf{P}\mathbf{Y}_{\mathcal{X}}' \,, \tag{6.28}$$

where columns of matrix $\bar{\mathbf{F}}_{\mathcal{X}}$ are vectors of first order sufficient statistics, $\bar{\mathbf{f}}$, extracted from all segments in training set (representing test segments) and where columns of matrix $\mathbf{Y}_{\mathcal{X}}$ are vectors of speaker factors extracted from all segments in training set (representing enrollment segments).

Now, the gradient could be used to optimize the objective function using standard gradient descent method. However, the widely adopted technique for MMI training GMMs of HMMs is Extended Baum-Welch [Schlüter, 2001] re-estimation, which has been shown to provide much faster convergence compared to gradient descent. In our case, the Extended Baum-Welch can not be adopted in straightforward way thanks to our more complicated model and simplified linear scoring. In [Schlüter, 2001] section 2.2.2., relation between Extended Baum-Welch and gradient descent update of parameters was pointed out showing that Extended Baum-Welch update of GMM mean vectors can be seen as gradient descent update with a specific learning rate used to update each parameter. Inspired by this relation, we propose to use similar learning rate specific to each row of matrix **V**. Specifically, we multiply the gradient $\nabla E(\mathbf{V})$ by diagonal matrix

$$\mathbf{L} = \eta \operatorname{diag}(\tilde{\mathbf{N}}_{\mathcal{X}}\mathbf{P}\mathbf{1})\boldsymbol{\Sigma}, \tag{6.29}$$

where columns of matrix $\tilde{\mathbf{N}}_{\mathcal{X}}$ are vectors of zero order sufficient statistics extracted from all segments in training set, η is parameter-independent learning rate, **1** is column vector of ones and diag(·) is an operator that converts vector into diagonal matrix. Finally, the matrix **V** is iteratively updated using the following formula:

$$\mathbf{V}^{\text{new}} = \mathbf{V}^{\text{old}} + \mathbf{L}\nabla E(\mathbf{V}) \,. \tag{6.30}$$

6.2.1 Experiment with no explicit channel compensation

The first system, which we used for experimenting with discriminative training of eigenvoices **V** was pure eigenvoice system (i.e., no **U** and no **D** matrices were considered in this case). A system based on relatively small UBM with only 512 components and 39 dimensional features was selected for this experiment. The system was first trained using maximum likelihood, then eigenvoice matrix **V** of size 19968×300 was retrained discriminatively using the procedure described in the previous section. We use ML trained **V** as the staring point for the discriminative training. The results are presented in Table 6.1. Comparing the first and the third line in the table, we can see that discriminative training provides substantial improvement in the performance. The improvement holds also when applying zt-normalization, though the normalization was not considered during the discriminative training.

As mentioned in the previous section, speaker factors **y** are always computed using the original ML trained model. In this case, it is the pure eigenvoice system with speaker factors **y** estimated without considering channel variability. The last line in the table

Table 6.1: Results of the 1st large scale experiment, on SRE 2006 all trials (det1).

EER[%]	No norm	ZT-norm
Generative **V**	15.44	11.42
Generative **V** and **U**	6.99	4.07
Discriminative **V**	7.19	5.06
Discriminative **V** with channel compensated **y**	6.80	4.81

Table 6.2: Results of the 2nd large scale experiment, on SRE 2006 all trials (det1).

EER[%]	No norm	ZT-norm
Generative **V** and **U**	6.99	4.07
Discriminative **V**, Generative **U**	6.00	3.87

shows results obtained with the same discriminatively trained pure eigenvoice system used for testing, where, however, factors **y** were obtained from ML trained system modeling the channel variability. The improved result suggests that good estimation of **y** not affected by channel variability may be important. Possibly, in the future, this could be also achieved by means of discriminative training, without explicitly modeling the channel variability.

The second line in the table shows performance of ML trained system making use of eigenchannels for both estimating speaker factors **y** and testing. We can see that discriminative training provides improvements comparable to the intersession variability modeling. However, improvement over the ML trained system is observed only in the first column of the third row, which corresponds to result without zt-normalization. When zt-norm is used, performance of the generative system is superior to the discriminatively trained one. Note, that zt-norm was not considered during discriminative training. Not having zt-norm incorporated in the discriminative training may force the training to concentrate on problem that can be easily solved by the normalization, which can lead to suboptimal result.

6.2.2 Experiment with ML trained eigenchannels

In the following experiment, the intersession variability is modeled using eigenchannel matrix **U** (200 eigenchannels). Otherwise, the system is the same as in the previous set of experiments. Again, the ML trained parameters are used as the starting point for the discriminative training of matrix **V**. Although we make use of **U** in this experiment, this matrix is not retrained discriminatively. As can be seen in Table 6.2, retraining the matrix **V** improves the results. Much higher improvement is, however, obtained without zt-norm. The probable reason is the same as explained in the previous paragraph.

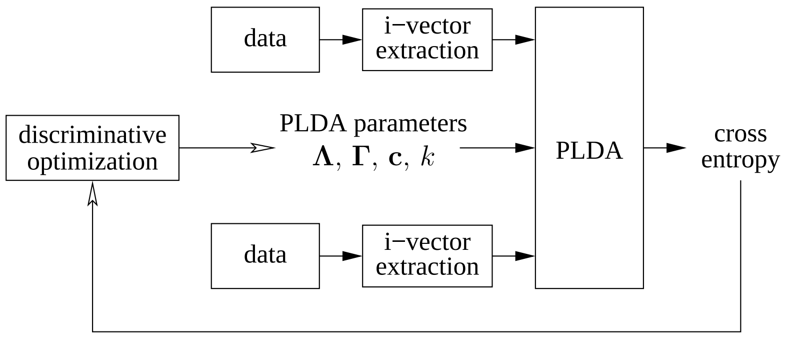

data
i−vector extraction
discriminative optimization
PLDA parameters
$\mathbf{\Lambda}$, $\mathbf{\Gamma}$, $\mathbf{c}$, k
PLDA
cross entropy
data
i−vector extraction

Figure 6.7: System setup for the discriminative training experiments.

6.3 Discriminative Training of PLDA

Let us now show how the discriminative classifier from Section 6.1 can be used in training the PLDA.

6.3.1 Efficient score and Gradient evaluation

We recall that the computation of a bilinear form $\mathbf{x}^T\mathbf{A}\mathbf{y}$ can be expressed in terms of the Frobenius inner product as $\mathbf{x}^T\mathbf{A}\mathbf{y} = \langle \mathbf{A}, \mathbf{x}\mathbf{y}^T\rangle = \text{vec}(\mathbf{A})^T\text{vec}(\mathbf{x}\mathbf{y}^T)$, where $\text{vec}(\cdot)$ stacks the columns of a matrix into a vector. Therefore, the log-likelihood (5.36) from Section 5.3.1 can be written as a dot product of a vector of weights $\mathbf{w}^T$, and an expanded vector $\boldsymbol{\varphi}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2)$ representing a trial:

$$\begin{aligned} s &= \mathbf{w}^T\boldsymbol{\varphi}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) \\ &= \begin{bmatrix} \text{vec}(\mathbf{\Lambda}) \\ \text{vec}(\mathbf{\Gamma}) \\ \mathbf{c} \\ k \end{bmatrix}^T \begin{bmatrix} \text{vec}(\boldsymbol{\phi}_1\boldsymbol{\phi}_2^T + \boldsymbol{\phi}_2\boldsymbol{\phi}_1^T) \\ \text{vec}(\boldsymbol{\phi}_1\boldsymbol{\phi}_1^T + \boldsymbol{\phi}_2\boldsymbol{\phi}_2^T) \\ \boldsymbol{\phi}_1 + \boldsymbol{\phi}_2 \\ 1 \end{bmatrix} . \end{aligned} \tag{6.31}$$

where the PLDA parameters $\mathbf{\Lambda}$, $\mathbf{\Gamma}$, $\mathbf{c}$, and k are defined in equations (5.37) through (5.40). Hence, we have obtained a generative generalized linear classifier [Bishop, 2006]. The advantage of this form is that we have expressed the parameters as one single vector $\mathbf{w}$ and this will be the form that we are going to optimize.

We have already defined the general gradient of the objective function w.r.t. the score in (6.20). To evaluate the gradient w.r.t. the parameters $\mathbf{w}$, we use the chain rule and evaluate the Jacobian matrix of the score w.r.t. $\mathbf{w}$ and use it in (6.20). The derivation is given as

$$\begin{aligned} \frac{\partial s}{\partial \mathbf{w}} &= \frac{\partial}{\partial \mathbf{w}}\mathbf{w}^T\boldsymbol{\varphi}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) \\ &= \boldsymbol{\varphi}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) . \end{aligned} \tag{6.32}$$

Given a trained classifier, we can obtain a verification score for a trial by forming the expanded vector $\boldsymbol{\varphi}(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2)$ and computing the dot product (6.31). However, as we have already seen, the same score can be obtained using the two original i-vectors $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2$ and using formula (5.36), which is both memory and computationally efficient. Now, consider two sets of i-vectors stored as columns of the matrices $\boldsymbol{\Phi}_e$ and $\boldsymbol{\Phi}_t$. For illustration, let us call these sets enrollment and test trials, although they play symmetrical roles in our scoring scheme. We can efficiently score each enrollment trial against each test trial and obtain the full matrix of scores as

$$\begin{aligned}\mathbf{S} \;=\; & 2\boldsymbol{\Phi}_e^T\boldsymbol{\Lambda}\boldsymbol{\Phi}_t \\ & +((\boldsymbol{\Phi}_e^T\boldsymbol{\Gamma}) \circ \boldsymbol{\Phi}_e^T)\mathbf{1}\mathbf{1}^T + \mathbf{1}\mathbf{1}^T(\boldsymbol{\Phi}_t \circ (\boldsymbol{\Gamma}\boldsymbol{\Phi}_t)) \\ & +\boldsymbol{\Phi}_e^T\mathbf{c}\mathbf{1}^T + \mathbf{1}\mathbf{c}^T\boldsymbol{\Phi}_t^T + k, \qquad (6.33)\end{aligned}$$

where $\circ$ denotes the Hadamard, or "entry-wise" product. Similarly, the naïve way of evaluating the Jacobian matrix would be to explicitly expand every training trial and then to apply equations (6.20) to (6.32). However, again taking into account the functional form for computing scores (5.36), the gradient can be evaluated much more efficiently without any need for explicit trial expansion. Let all the i-vectors, which we have available for training, be stored in columns of a matrix $\boldsymbol{\Phi}$. Now consider forming a training trial using every possible pair of i-vectors from this matrix. Let s_{ij} be the score for the trial formed by the i-th and j-th columns of $\boldsymbol{\Phi}$ calculated using the parameters $\mathbf{w}$ for which we wish to evaluate the gradient. Let t_{ij} and α_{ij} be the corresponding label and trial weight, respectively. Further, let d_{ij} be the corresponding derivation of loss function $E(t_{ij}s_{ij})$ w.r.t. the score s_{ij} given in (6.21) or (6.22) depending on the loss function used. The gradient can now be efficiently evaluated as

$$\nabla E(\mathbf{w}) = \begin{bmatrix}\nabla_\Lambda L \\ \nabla_\Gamma L \\ \nabla_c L \\ \nabla_k L\end{bmatrix} = \begin{bmatrix} 2 \cdot \operatorname{vec}\left(\boldsymbol{\Phi}\mathbf{G}\boldsymbol{\Phi}^T\right) \\ 2 \cdot \operatorname{vec}\left(\boldsymbol{\Phi}[\boldsymbol{\Phi}^T \circ (\mathbf{G}\mathbf{1}\mathbf{1}^T)]\right) \\ 2 \cdot \mathbf{1}^T[\boldsymbol{\Phi}^T \circ (\mathbf{G}\mathbf{1}\mathbf{1}^T)] \\ \mathbf{1}^T\mathbf{G}\mathbf{1}\end{bmatrix} + \lambda\mathbf{w} \qquad (6.34)$$

where elements of matrix $\mathbf{G}$ are $g_{ij} = d_{ij}\alpha_{ij}$.

6.3.2 Numerical Optimization

The numerical optimization is based on the iterative "trust-region Newton-conjugate-gradient" (CG) method, as described in [Lin et al., 2008, Nocedal and Wright, 2006]. The "trust-region" (or restricted-step) refers to a spherical region in the space of the optimized parameters, which serves as an update limit in case the optimization step goes out of the trusty limits. The region is dynamically changed based on certain heuristics from each previous update step.

The optimization itself is based on the Newton's optimization method, where the gradient and the (inverted) Hessian are used for faster convergence. However, the update step of the Newton-CG does not directly involve the inversion of the Hessian, nor it has to evaluate the Hessian itself; instead, the inversion is computed using the conjugate-gradient method, and the update step is effectively computed via a Hessian-vector multiplication. Therefore, the implementation and application of the algorithm can be efficient in that one only needs to provide a function which efficiently computes such multiplication. Evaluating

Table 6.3: Normalized DCF_{new}, $DCF_{old}$and EER for the extended condition 5 (tel-tel) from the NIST SRE 2010 evaluation.

	Female Set			Male Set			Pooled		
System	DCF_{new}	DCF_{old}	EER	DCF_{new}	DCF_{old}	EER	DCF_{new}	DCF_{old}	EER
PLDA	0.40	0.15	3.57	0.42	0.13	2.86	0.41	0.14	3.23
LR	0.40	0.12	2.94	0.39	0.10	2.22	0.40	0.11	2.62
SVM	0.39	0.11	2.35	0.31	0.08	1.55	0.37	0.10	1.94
HT-PLDA	0.34	0.11	2.22	0.33	0.08	1.47	0.34	0.10	1.88

the Hessian-vector product function can still be rather complex; therefore, we employed a numerical approximation. The 'complex step differentiation' [Shampine, 2007] was tested first, however, due to code optimization purposes, real-step numerical approximation—where the product is expressed in terms of two close-enough gradient vectors—was used.

The algorithm was taken off-the-shelf as a part of the BOSARIS toolkit [Brümmer and de Villiers, 2010]. The toolkit allows for building the cross-entropy objective function as a composition of atomic functions represented by blocks of MATLAB routines. Each such block provides the atomic function evaluation, as well as its vector-Jacobian product to allow for back-propagation of the gradient for the evaluation of the complete composite gradient.

6.3.3 Experimental Setup

The i-vector extractor and the baseline PLDA system is taken from the ABC system submitted to NIST SRE 2010 evaluation [Brümmer et al., 2010]. The i-vector extractor uses 60-dimensional cepstral features and a 2048-component full covariance GMM. The UBM and i-vector extractor are trained on NIST SRE 2004, 2005 and 2006, Switchboard and Fisher data. All PLDA systems and discriminative classifiers are trained using 400 dimensional i-vectors extracted from 21663 segments from 1384 female speakers and 16969 segments from 1051 male speakers from NIST SRE 2004, NIST SRE 2005, NIST SRE 2006, Switchboard II Phases 2 and 3, and Switchboard Cellular Parts 1 and 2.

6.3.4 Results

Table 6.3 presents results for the extended condition 5 (tel-tel) from NIST SRE 2010 evaluation. The reported numbers are Equal Error Rate (EER) and normalized minimum Decision Cost Functions for the two operating points as defined by NIST for the SRE 2008 (DCF_{old}) and SRE 2010 (DCF_{new}) evaluations as described in Section 2.3.

The system denoted as PLDA, which serves as the baseline, is based on a generatively trained PLDA model with a 90-dimensional speaker variability subspace [Brümmer et al., 2010]. On telephone data, this configuration was found to give the best DCF_{new}(focusing on low FA rates), which was the primary performance measure in the NIST SRE 2010 evaluation. As a trade-off, the system gives somewhat poorer performance at the $\mathrm{DCF}_{\mathrm{old}}$and EER.

The system denoted as LR is the discriminative linear classifier, where parameters were initialized from the baseline system using (5.37) through (5.40) and retrained to optimize the logistic regression objective function. Trust-region Newton-CG method, as described in Section 6.3.2, was used. No regularization was used in this case. Significant improvements compared to the baseline can be observed, especially at $\mathrm{DCF}_{\mathrm{old}}$ and EER.

For comparison with the con-current state-of-the art techniques, the presented results include the SVM discriminative training [Cumani et al., 2011] (as a parallel work on discriminative PLDA), and the HTPLDA, which was the state-of-the-art generative approach.

With the SVM-based classifier, the improvement was even larger than the LR. 10%, 30% and 40% relative improvements over the baseline were obtained for $\mathrm{DCF}_{\mathrm{new}}$, $\mathrm{DCF}_{\mathrm{old}}$ and EER respectively. The improvements over the LR system can probably be attributed mainly to the presence of the regularization term. Often, SVM classifiers are trained using a solver to the dual problem, where a Gram matrix needs to be evaluated. The Gram matrix is a matrix comprising dot products between every pair of training examples, which are the trials in our case. Since it was decided to construct a training trial for every pair of i-vectors, the size of the Gram matrix would be unmanageably large (the number of training i-vectors to the 4th power). Therefore, a linear SVM was trained by again solving the primal problem using a solver [Teo et al., 2007], which makes use of the efficient evaluation of gradient. To make SVM regularization effective, it was found that it is necessary to first normalize input i-vectors using within-class covariance normalization (WCCN) [Dehak et al., 2010], i.e. to normalize i-vectors to have identity within-class covariance matrix.

Finally, for comparison, we also include results with Heavy-tailed PLDA (HT-PLDA) [Kenny, 2010], which gave the best results obtained with the same set of training and test i-vectors. In heavy-tailed PLDA, speaker and intersession variability are modeled using Student's t, rather than Gaussian distributions. In the compared system, the dimensionality of i-vectors was first reduced from 400 to 120 and the final vectors were modeled with full-rank speaker and intersession subspaces. Nevertheless, the price paid for the excellent results obtained with heavy-tailed PLDA is the very computationally demanding score evaluation. As we can see, competitive results can be obtained with the discriminatively trained models, for which the score evaluation is several orders of magnitude faster.

It is worth mentioning that it was later observed that length normalization plays an important role in this set of experiments. Not only it makes the Gaussian PLDA competitive to the HT-PLDA [Garcia-Romero, 2011], but also it makes the discriminative PLDA training ineffective.

6.4 Discriminative Training of i-vector Extractor

Let us first describe the system setup in order to understand the further description of the problem. The speaker recognition pipeline is shown in Figure 6.8. The data comes in

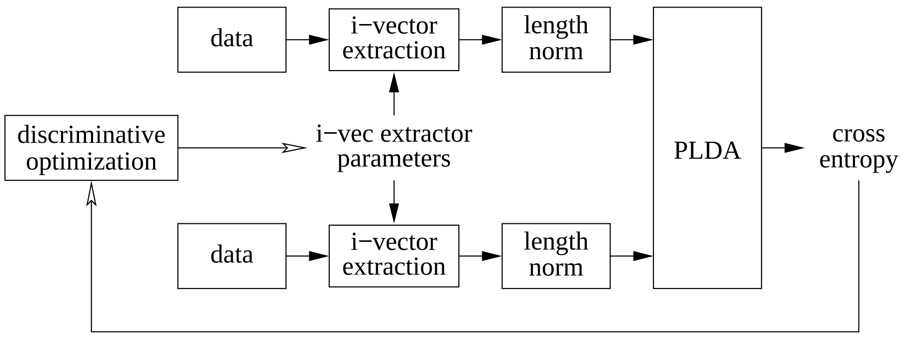


Figure 6.8: System setup for the discriminative training experiments.

the form of zero- and first-order statistics. It is then processed by the i-vector extractor (in one of the two forms studied in Section 5.1.3 and Section 5.4.2). Length normalization (as described in Section 5.3) is then performed. A pair of such i-vectors is then processed by PLDA (Section 5.3) and a single score per trial is further passed to the cross-entropy error function, as described in (6.13). As seen in the figure, we need to optimize the last box, i.e., the cross-entropy, with respect to the parameters of the i-vector extractor. There is no closed-form solution to this problem since the cross-entropy is a multiply-composed function of the i-vector extractor parameters. Therefore, numerical-optimization needs to be used.

We discriminatively optimize two kinds of i-vector extractor. In the first case, the traditional extraction—as described in Section 5.1.3—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-2" extraction (as described in Section 5.4.2) 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*.

6.4.1 Gradient Evaluation

The gradient of the objective function has been defined in Section 6.1.1. We will now subsequently apply the chain rule to express the gradient in terms of the optimized parameters. 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 the gradient as a sum of two gradients

$$\frac{\partial s_n}{\partial \boldsymbol{\theta}} = \frac{s(\bar{\boldsymbol{\phi}}_1, \bar{\boldsymbol{\phi}}_2)}{\partial \bar{\boldsymbol{\phi}}_1} \frac{\partial \bar{\boldsymbol{\phi}}_1}{\partial \boldsymbol{\theta}} + \frac{s(\bar{\boldsymbol{\phi}}_1, \bar{\boldsymbol{\phi}}_2)}{\partial \bar{\boldsymbol{\phi}}_2} \frac{\partial \bar{\boldsymbol{\phi}}_2}{\partial \boldsymbol{\theta}} \; . \tag{6.35}$$

From the bilinear form of PLDA as given by (5.36), knowing that $\boldsymbol{\Lambda}$ and $\boldsymbol{\Gamma}$ are symmetrical, we can derive

$$\frac{s(\bar{\boldsymbol{\phi}}_1, \bar{\boldsymbol{\phi}}_2)}{\partial \bar{\boldsymbol{\phi}}_1} = 2\boldsymbol{\phi}_2'\boldsymbol{\Lambda} + 2\boldsymbol{\phi}_1'\boldsymbol{\Gamma} + \mathbf{c} \; . \tag{6.36}$$

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 as defined by (5.26)

$$\frac{\partial \bar{\boldsymbol{\phi}}}{\partial \boldsymbol{\theta}} = \frac{\partial \bar{\boldsymbol{\phi}}}{\partial \boldsymbol{\phi}} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{\theta}} \tag{6.37}$$

where

$$\frac{\partial \bar{\boldsymbol{\phi}}}{\partial \boldsymbol{\phi}} = \frac{1}{\|\boldsymbol{\phi}\|} \left(\mathbf{I} - (\bar{\boldsymbol{\phi}} \bar{\boldsymbol{\phi}}') \right) . \tag{6.38}$$

Finally, we derive through the i-vector extractor. This is slightly tricky as the derivation of a vector w.r.t. a matrix $\frac{\boldsymbol{\phi}}{\boldsymbol{\theta}}$ from (6.37) gives a third-order tensor. The naïve way could be to represent $\boldsymbol{\theta}$ as a vector and derive w.r.t each coefficient. However, we can make use of matrix differentials.

Let us note that we can construct our optimization trial set $\mathcal{X}$ by pairing all M training segments with all and masking them using the α_x constant. It is trivial to express the cross-entropy E as a function of the complete set of M i-vectors $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1 \cdots \boldsymbol{\phi}_M]$, where each trial will be given as a pair of indexes to the training segments ij:

$$E(\boldsymbol{\Phi}) = \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_{ij} E_{LR} \left(t_{ij} s(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j) \right) . \tag{6.39}$$

With the given formulas for derivatives, it is also straightforward to express the gradient $\frac{\partial E(\boldsymbol{\Phi})}{\partial \boldsymbol{\Phi}}$:

$$\frac{\partial E(\boldsymbol{\Phi})}{\partial \boldsymbol{\Phi}} = \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_{ij} \frac{\partial E_{LR}}{s_{ij}} \left(\frac{\partial s(\bar{\boldsymbol{\phi}}_i, \bar{\boldsymbol{\phi}}_j)}{\partial \bar{\boldsymbol{\phi}}_i} \frac{\partial \bar{\boldsymbol{\phi}}_i}{\partial \boldsymbol{\phi}_i} + \frac{\partial s(\bar{\boldsymbol{\phi}}_i, \bar{\boldsymbol{\phi}}_j)}{\partial \bar{\boldsymbol{\phi}}_j} \frac{\partial \bar{\boldsymbol{\phi}}_j}{\partial \boldsymbol{\phi}_j} \right) . \tag{6.40}$$

To derive through the i-vector extractor, we will make use of the chain rule for differentials, where the following holds:

$$dE = \sum_{ij} \frac{\partial E}{\partial \phi_{ij}} d\phi_{ij} = \sum_{kl} \frac{\partial E}{\partial \theta_{kl}} d\theta_{kl} , \tag{6.41}$$

i.e., the output differential of any stage is always determined by the gradient of that stage and the input differential, which has already been discussed earlier in Section 6.1.1. Looking at (6.41), we can also say that if we are able to evaluate the gradient $\frac{\partial E}{\partial \phi_{ij}}$ and we can express $d\boldsymbol{\Phi}$ in terms of $d\boldsymbol{\theta}$, we are also able to find $\frac{\partial E}{\partial \boldsymbol{\theta}}$. We can do that by making use of the matrix differentials. For the full i-vector extractor, the differential for j-th column of $d\boldsymbol{\Phi}$ is given as

$$d\boldsymbol{\phi}_j = -\mathbf{L}_j^{-1} d\mathbf{L}_j \mathbf{L}_j^{-1} \mathbf{T}' \mathbf{f}_j + \mathbf{L}_j^{-1} d\mathbf{T}' \mathbf{f}_j , \tag{6.42}$$

where

$$d\mathbf{L}_j = \sum_c N_j^{(c)} \left(d\mathbf{T}^{(c)\prime} \mathbf{T}^{(c)} + \mathbf{T}^{(c)\prime} d\mathbf{T}^{(c)} \right) . \tag{6.43}$$

In the case of the simplified i-vector extractor, the corresponding differentials w.r.t. the matrices **T**, **G**, and **V** are given respectively as

$$d\boldsymbol{\phi}_{\mathbf{T}j} = \mathbf{G}\tilde{\mathbf{L}}_j^{-1}\mathbf{G}' d\mathbf{T}'\mathbf{f}_j \tag{6.44}$$

$$d\boldsymbol{\phi}_{\mathbf{G}j} = \left(d\mathbf{G}\tilde{\mathbf{L}}_j^{-1}\mathbf{G}' + \mathbf{G}\tilde{\mathbf{L}}_j^{-1}d\mathbf{G}'\right)\mathbf{T}'\mathbf{f}_j \tag{6.45}$$

$$d\boldsymbol{\phi}_{\mathbf{V}j} = -\mathbf{G}\tilde{\mathbf{L}}_j^{-1}\mathrm{Diag}(d\mathbf{V}\mathbf{n}_j)\tilde{\mathbf{L}}_j^{-1}\mathbf{G}'\mathbf{T}'\mathbf{f}_j \tag{6.46}$$

where $\tilde{\mathbf{L}}$ is defined in (5.46). We find the gradient $\frac{\partial E(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ by substituting one of the $d\boldsymbol{\phi}$ from the above catalogue to (6.41). We can do that with the help of the $\mathrm{tr}(\cdot)$ function which allows to rewrite (6.41) as

$$\mathrm{tr}\left(\frac{\partial E(\boldsymbol{\Phi})}{\partial \boldsymbol{\Phi}} d\boldsymbol{\Phi}\right) = \mathrm{tr}\left(\frac{\partial E(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} d\boldsymbol{\theta}\right). \tag{6.47}$$

The tr function allows for rearrangements of the matrix products and once we have expressed the argument in such a way that $d\boldsymbol{\theta}$ is the right-most term of the product, the left side of it will be the solution as in r.h.s. of (6.47). In the case of the full i-vector extractor, the derivative can be expressed as

$$\frac{\partial E(\mathbf{T})}{\partial \mathbf{T}} = \sum_{j=1}^{M}\left[-\left(\mathbf{L}_j^{-1}\frac{\partial E}{\partial \boldsymbol{\phi}_j'}\boldsymbol{\phi}_j' + \boldsymbol{\phi}_j\frac{\partial E}{\partial \boldsymbol{\phi}_j}\mathbf{L}_j^{-1}\right)\mathbf{T}'\mathbf{N}_j + \mathbf{L}_j^{-1}\frac{\partial E}{\partial \boldsymbol{\phi}_j}\mathbf{f}_j\right]. \tag{6.48}$$

For the simplified i-vector extraction, the derivatives of the parameters are

$$\frac{\partial E(\mathbf{T})}{\partial \mathbf{T}} = \sum_{j=1}^{M}\mathbf{f}_j\frac{\partial E}{\partial \boldsymbol{\phi}_j}\mathbf{G}\hat{\mathbf{L}}_j^{-1}\mathbf{G}' \tag{6.49}$$

$$\frac{\partial E(\mathbf{G})}{\partial \mathbf{G}} = \sum_{j=1}^{M}\hat{\mathbf{L}}_j^{-1}\mathbf{G}'\left(\mathbf{T}'\mathbf{f}_j\frac{\partial E}{\partial \boldsymbol{\phi}_j} + \frac{\partial E}{\partial \boldsymbol{\phi}_j'}\mathbf{f}_j'\mathbf{T}\right) \tag{6.50}$$

$$\frac{\partial E(\mathbf{V})}{\partial \mathbf{V}} = \sum_{j=1}^{M}-\mathbf{n}_j\left(\frac{\partial E}{\partial \boldsymbol{\phi}_j}\mathbf{G}' \circ \mathbf{f}_j'\mathbf{T}\mathbf{G}\hat{\mathbf{L}}_j^{-2}\right), \tag{6.51}$$

where the $\circ$ stands for the Hadamard product. The gradients $\frac{\partial E}{\partial \boldsymbol{\phi}_j}$ are simply the columns of (6.40).

6.4.2 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, nd]. The recognition accuracy is given in terms of equal error rate (EER), and the normalized DCF as defined in both NIST 2010 SRE (DCF$_{\mathrm{new}}$) and the previous SRE evaluations (DCF$_{\mathrm{old}}$).

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 5.4.2.

As described in Section 5.3, 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.

Results

The numerical optimization of the parameters was performed in MATLAB using the optimization and differentiation tools in the BOSARIS Toolkit as mentioned in Section 6.3.2. Target prior $p(\mathcal{H}_1)$ was set to 0.001 accroding to NIST2010 requirement (see Table 2.1).

First, the full i-vector extractor was tested. The first experiments were done with initializing matrix **T** to random or to zeros. Even though the error function was decreasing, the EER on the test set was close to random even after 20 iterations and no sign of improvement was observed. The situation changed when initialized from the ML solution. 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. Figure 6.9a shows the plot of the objective function and EER throughout the iterations when no regularizer was used. We see that even though

Table 6.4: *Comparison of ML and discriminatively trained full i-vector extractors in terms of normalized DCFs and EER*

	$\mathrm{DCF}_{\mathrm{new}}$	$\mathrm{DCF}_{\mathrm{old}}$	EER
ML	0.6678	0.2200	4.74
discriminative	0.6548	0.2122	4.26

Table 6.5: *Comparison of ML and discriminatively trained simplified i-vector extractors in terms of normalized DCFs and EER*

	$\mathrm{DCF}_{\mathrm{new}}$	$\mathrm{DCF}_{\mathrm{old}}$	EER
ML	0.7496	0.2710	6.18
discriminative	0.6691	0.2403	5.41

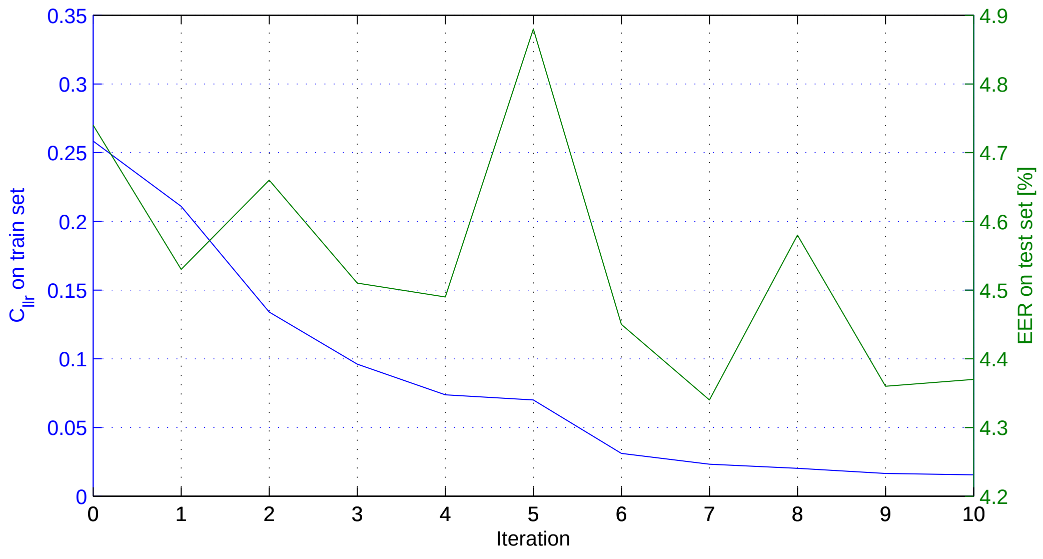
the overall trend is decreasing EER, the optimization is unstable. Figure 6.9b—on the other hand—shows the progression of the optimization when the described regularizer was used. Plain L2 regularizer (omitting the $\boldsymbol{\theta}_{\mathrm{ML}}$ term in (6.13)) was also tested; however, it turned out that together with good initialization, the discriminative training works only as a "fine-tuner" of the initial parameters.

Retraining the PLDA parameters (in generative fashion) between the iterations was also tested, but never improved the results and usually increased the error function.

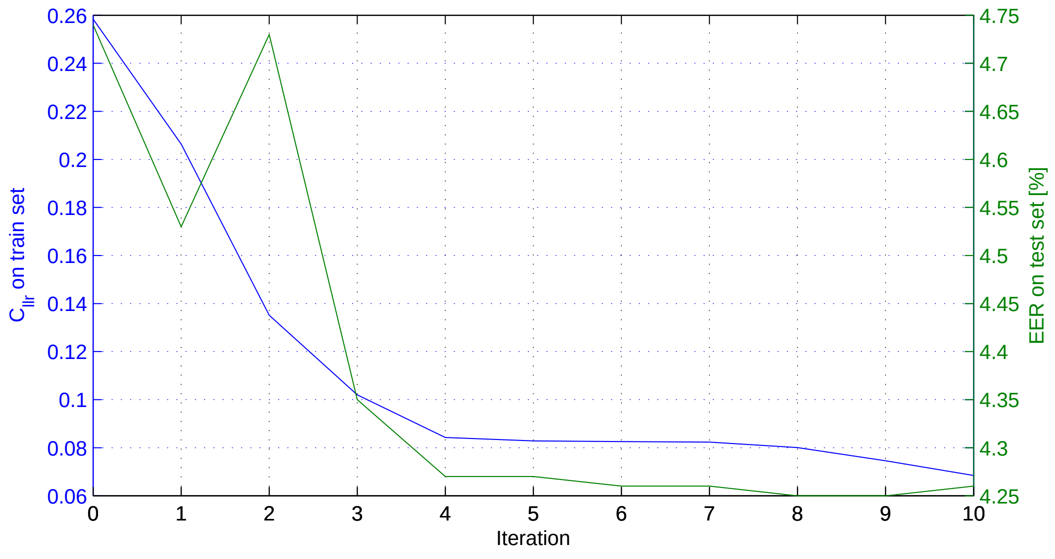
In the case of the simplified version, the matrices $\mathbf{G}$ and $\mathbf{T}$ were optimized subsequently. It was found, however, that even though optimizing $\mathbf{V}$ kept on decreasing the error function, it would always decrease the recognition performance on the test set. The regularization term was used in the same way as in the case of the full i-vector extractor.

It turned out that 10 iteration steps of the optimization algorithm gave stable results in both cases when regularizer term was used.

Table 6.4 gives the comparison of the system based on the generative extractor and discriminatively retrained one. We see that there is about 10% relative improvement in EER, however moving towards low-false alarm operating points, the improvement gets smaller. In the case of the simplified i-vector extractor, the improvement is more apparent—see Table 6.5 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 6.9: 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 7

Conclusions

7.1 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.

7.2 Future Work

7.2.1 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.

7.2.2 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.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Brümmer, 2004] Brümmer, N. (2004). Spescom DataVoice NIST 2004 system description. In *Proc. NIST Speaker Recognition Evaluation 2004*, Toledo, Spain.

[Brümmer, 2009a] Brümmer, N. (2009a). The EM algorithm and minimum divergence: Technical report, Agnitio Research, South Africa. `https://sites.google.com/site/nikobrummer/EMandMINDIV.pdf`.

[Brümmer, 2009b] Brümmer, N. (2009b). EM for JFA: Technical report, Agnitio Research, South Africa. `https://sites.google.com/site/nikobrummer/EMforJFA.pdf`.

[Brümmer, 2010] Brümmer, N. (2010). EM for PLDA: Technical report, Agnitio Research, South Africa. `https://sites.google.com/site/nikobrummer/EMforPLDA.pdf`.

[Brümmer et al., 2010] Brümmer, N., Burget, L., Kenny, P., Matějka, P., de Villiers, E., Karafiát, M., Kockmann, M., Glembek, O., Plchot, O., Baum, D., and Senoussauoi, M. (2010). ABC system description for NIST SRE 2010. In *Proc. of NIST 2010 Speaker Recognition Evaluation, Brno, Czech Republic*, pages 1–20.

[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 du Preez, 2006] Brümmer, N. and du Preez, J. (2006). Application-independent evaluation of speaker detection. *Computer Speech & Language*, 20(2-3):230–275.

[Brümmer and van Leeuwen, 2006] Brümmer, N. and van Leeuwen, D. (2006). On calibration of language recognition scores. In *Proceedings of Speaker and Language Recognition Workshop, 2006. IEEE Odyssey 2006: The*, pages 1 –8.

[Burget, 2004] Burget, L. (2004). *Complementarity of Speech Recognition Systems and System Combination.* PhD thesis, Brno University of Technology.

[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., 2007] Burget, L., Matejka, P., Schwarz, P., Glembek, O., and Cernocky, J. (2007). Analysis of feature extraction and channel compensation in GMM speaker recognition system. *IEEE Transactions on Audio, Speech and Language Processing*, 15(7):1979–1986.

[Burget et al., 2009] Burget, L., Matějka, P., Hubeika, V., and Černocký, J. (2009). Investigation into variants of joint factor analysis for speaker recognition. In *Proc. Interspeech 2009*, number 9, pages 1263–1266.

[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., 2006a] Campbell, W., Gleason, T., Navrátil, J., Reynolds, D., Shen, W., Singer, E., and Torres-Carrasquillo, P. (2006a). Advanced language recognition using cepstra and phonotactics: MITLL system performance on the NIST 2005 language recognition evaluation. In *Proceedings of IEEE Odyssey 2006: The Speaker and Language Recognition Workshop*, San Juan, Puerto Rico.

[Campbell et al., 2006b] Campbell, W., Sturim, D., Reynolds, D., and Solomonoff, A. (2006b). 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.

[Cumani, 2012] Cumani, S. (2012). *Speaker and Language Recognition Techniques.* PhD thesis, Politecnico di Torino.

[Cumani et al., 2011] Cumani, S., Brümmer, N., Burget, L., and Laface, P. (2011). Fast discriminative speaker verification in the i-vector space. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 4852 –4855.

[Davis and Mermelstein, 1980] Davis, S. B. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4).

[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.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38.

[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.

[Douglas Reynolds, 2000] Douglas Reynolds, Thomas F. Quatieri, R. B. D. (2000). Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, pages 19–41.

[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, 1999a] Gales, M. (1999a). Cluster adaptive training of hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 8:417–428.

[Gales, 1999b] Gales, M. (1999b). 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.

[Graff et al., 2001] Graff, D., Miller, D., and Walker, K. (2001). Switchboard cellular part 1 audio. *Linguistic Data Consortium, Philadelphia.*

[Graff et al., 2002] Graff, D., Miller, D., and Walker, K. (2002). Switchboard-2 phase III. *Linguistic Data Consortium, Philadelphia.*

[Graff et al., 2004] Graff, D., Miller, D., and Walker, K. (2004). Switchboard cellular part 2 audio. *Linguistic Data Consortium, Philadelphia.*

[Graff et al., 1999] Graff, D., Walker, K., and Canavan, A. (1999). Switchboard-2 phase II. *Linguistic Data Consortium, Philadelphia.*

[Hadid and McCool, 2010] Hadid, A. and McCool, C. (2010). Description of MOBIO database. `https://www.idiap.ch/dataset/mobio/description-1`.

[Hatch et al., 2006] Hatch, A. O., Kajarekar, S., and Stolcke, A. (2006). Within-Class Covariance Normalization for SVM-based speaker recognition. In *Proc. ICSLP, Pittsburgh, USA*, pages 1471–1474.

[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). Disentangling speaker and channel effects in speaker verification. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, volume 1, pages I – 37–40 vol.1.

[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., 2003] Kenny, P., Mihoubi, M., and Dumouchel, P. (2003). New map estimators for speaker recognition. In *INTERSPEECH*.

[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.

[Kuhn et al., 1998] Kuhn, R., Nguyen, P., Junqua, J. C., Goldwasser, L., Niedzielski, N., Fincke, S., Field, K., and Contolini, M. (1998). *Eigenvoices for speaker adaptation*.

[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.

[Martin et al., 1997] Martin, A., Doddington, G., Kamm, T., Ordowski, M., and Przybocki, M. (1997). The DET Curve in Assessment of Detection Task Performance. In *Proc. Eurospeech '97*, pages 1895–1898, Rhodes, Greece.

[Martin and Przybocki, 2004] Martin, A. and Przybocki, M. (2004). 2004 NIST speaker recognition evaluation. `http://www.itl.nist.gov/iad/mig//tests/sre/2004`.

[Matějka, 2009] Matějka, P. (2009). *Phonotactic and Acoustic Language Recognition*. PhD thesis, Brno University of Technology.

[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.

[Minka, 1998] Minka, T. (1998). Expectation-maximization as lower bound maximization. `http://research.microsoft.com/en-us/um/people/minka/papers/matrix/minka-matrix.pdf`.

[Navrátil et al., 2003] Navrátil, J., Jin, Q., Andrews, W., and Campbell, J. (2003). Phonetic speaker recognition using maximum-likelihood binary-decision tree models. In *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Hong Kong.

[NIST, 2005] NIST (2005). The NIST year 2005 speaker recognition evaluation plan.

[NIST, 2006] NIST (2006). The NIST year 2006 speaker recognition evaluation plan. `http://www.itl.nist.gov/iad/mig//tests/sre/2006`.

[NIST, 2008] NIST (2008). The NIST year 2008 speaker recognition evaluation plan. `http://www.itl.nist.gov/iad/mig//tests/sre/2008`.

[NIST, 2010] NIST (2010). The NIST year 2010 speaker recognition evaluation plan. `http://www.itl.nist.gov/iad/mig//tests/sre/2010`.

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

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

[Openshaw and Masan, 1994] Openshaw, J. and Masan, J. (1994). On the limitations of cepstral features in noise. In *Proc. ICASSP 1994*, Adelaide, SA, Australia.

[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.

[Pešán, 2011] Pešán, J. (2011). Speaker recognition on mobile phone. Master's thesis, Brno University of Technology.

[Povey, 2004] Povey, D. (2004). *Discriminative Training for Large Vocabulary Speech Recognition*. PhD thesis, Cambridge University.

[Povey et al., 2011] Povey, D., Burget, L., Agarwal, M., Akyazi, P., Kai, F., Ghoshal, A., Glembek, O., Goel, N., Karafiát, M., Rastrow, A., Rose, R. C., Schwarz, P., and Thomas, S. (2011). The subspace gaussian mixture model—structured model for speech recognition. *Computer Speech and Language*, 25(2):404 – 439.

[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*.

[Rabiner and Juang, 1993] Rabiner, L. and Juang, B.-H. (1993). *Fundamentals of speech recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Reynolds, 2002] Reynolds, D. (2002). Automatic speaker recognition – acoustics and beyond. JHU SW'02 Tutorial.

[Reynolds, 2003] Reynolds, D. (2003). Channel robust speaker verification via feature mapping. In *Proceedings of ICASSP '03. 2003 IEEE International Conference on*, volume 2, pages II – 53–6 vol.2.

[Schlüter, 2001] Schlüter, R. (2001). Comparison of discriminative training criteria and optimization methods for speech recognition. *Speech Communication*, 34(3):287–310.

[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.

[Teo et al., 2007] Teo, C., Smola, A., Vishwanathan, S. V., and Le, Q. V. (2007). A scalable modular convex solver for regularized risk minimization. In *Proc. KKD*, pages 727–736.

[Thyes et al., 2000] Thyes, O., Kuhn, R., Nguyen, P., and Junqua, J.-C. (2000). Speaker identification and verification using eigenvoices. In *INTERSPEECH*, pages 242–245.

[Torres-Carrasquillo et al., 2002] Torres-Carrasquillo, P. A., Singer, E., Kohler, M. A., and Deller, J. R. (2002). Approaches to language identification using gaussian mixture models and shifted delta cepstral features. In *Proc. ICSLP 2002*, pages 89–92.

[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.

[Vogt et al., 2005] Vogt, R., Baker, B., and Sridharan, S. (2005). Modelling session variability in text-independent speaker verication. In *Proc. Eurospeech*, pages 3117–3120, Lisbon, Portugal.

[Wikipedia, nd] Wikipedia (nd). Mixture model. `http://en.wikipedia.org/wiki/Mixture_model`.

[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.

Appendix A

A.1 Log-Likelihood with Hidden Variables

Let us collectively denote x as all observed variables (usually data) and y as all hidden variables, where $y \in \mathcal{Y}$. Let us assume that the observed values were generated by a model parametrized by θ. Then the likelihood of the data x is given as

$$p(x|\theta) = \int_{y \in \mathcal{Y}} p(x, y|\theta) \, \mathrm{d}y \; . \tag{A.1}$$

Let us introudce a distribution over the hidden variables $q(y)$. For any arbitrary choice of $q(y)$, the following holds (with proof later):

$$\log p(x|\theta) = \mathcal{L}(q(y), \theta) + \mathrm{D}_{\mathrm{KL}}(q(y) \| p(y|x, \theta)) \tag{A.2}$$

where

$$\mathcal{L}(q, \theta) = \int_{\mathcal{Y}} q(y) \log \frac{p(x, y|\theta)}{q(y)} \, \mathrm{d}y \tag{A.3}$$

$$\mathrm{D}_{\mathrm{KL}}(q(y) \| p(y|x, \theta)) = \int_{\mathcal{Y}} q(y) \log \frac{q(y)}{p(y|x, \theta)} \, \mathrm{d}y \; , \tag{A.4}$$

where the Kullback-Leibler divergence $\mathrm{D}_{\mathrm{KL}}(q(y) \| p(y|x, \theta))$ is always non-negative (by definition) and null iif $q(y) = p(y|x, \theta)$. This implies that $\mathcal{L}(q, \theta)$ is a *lower-bound* to the true log-likelihood for arbitrary choice of q. The proof for (A.2) is given by the following equality:

$$\log p(x|\theta) = \underbrace{\int_y q(y)\,\mathrm{d}y}_{1} \log p(x|\theta) \tag{A.5}$$

$$= \int_y q(y) \log p(x|\theta)\,\mathrm{d}y \tag{A.6}$$

$$= \int_y q(y) \log \frac{p(x,y|\theta)}{p(y|x,\theta)}\,\mathrm{d}y \tag{A.7}$$

$$= \int_y q(y) \log \frac{p(x,y|\theta)q(y)}{p(y|x,\theta)q(y)}\,\mathrm{d}y \tag{A.8}$$

$$= \int_y q(y) \log \frac{p(x,y|\theta)}{q(y)}\,\mathrm{d}y + \int_y q(y) \log \frac{q(y)}{p(y|x,\theta)}\,\mathrm{d}y \tag{A.9}$$

$$= \mathcal{L}(q,\theta) + \mathrm{D}_{\mathrm{KL}}(q(y)\|p(y|x,\theta))\ , \tag{A.10}$$

which prooves (A.2). Let us further expand the formula for diffent purposes

$$= \int_y q(y) \log \frac{p(x|y,\theta)p(y|\theta)}{q(y)}\,\mathrm{d}y + \mathrm{D}_{\mathrm{KL}}(q(y)\|p(y|x,\theta)) \tag{A.11}$$

$$= \int_y q(y) \log p(x|y,\theta)\,\mathrm{d}y - \int_y q(y) \log \frac{q(y)}{p(y|\theta)}\,\mathrm{d}y + \mathrm{D}_{\mathrm{KL}}(q(y)\|p(y|x,\theta)) \tag{A.12}$$

$$= \mathrm{E}_{q(y)}\left[\log p(x|y,\theta)\right] - \mathrm{D}_{\mathrm{KL}}(q(y)\|p(y|\theta)) + \mathrm{D}_{\mathrm{KL}}(q(y)\|p(y|x,\theta)) \tag{A.13}$$

A.2 The EM Algorithm in General

The expectation-maximization algorithm is a general iterative technique for finding ML estimates of parameters of probabilistic models with hidden variables [Dempster et al., 1977, Minka, 1998, Bishop, 2006, Brümmer, 2009a]. Let us use the notation from Section A.1.

EM is usually chosen when direct maximization of $p(x|\theta)$ is supposed to be difficult but $p(x,y|\theta)$ is signifficantly easier. The algorithm is based on the fact that $\mathcal{L}(q(y),\theta)$ in (A.2) is a lower-bound to the true log-likelihood and that by choosing proper $q(y)$ and maximizing $\mathcal{L}(q(y),\theta)$, we never decrease the log-likelihood. The algorithm then iterates between the E-step—in which the *auxilary function* is constructed via the choices of $q(y)$—and the M-step which maximizes this auxilary function

The E-step

The E-step is based on constructing an auxilary function $\mathcal{Q}(\theta,\theta_0)$—where θ_0 is some initial guess of the parameters—from $\mathcal{L}(q(y),\theta)$ by choosing appropriate $q(y)$ function independent of the optimized parameters θ. It is usual for $\mathcal{Q}$ (although not necessary) to touch the true log-likelihood function in the current estimate of the parameters θ_0, i.e. $\mathcal{Q}(\theta_0,\theta_0) = \log p(x|\theta_0)$. The solution is in setting

$$q(y) = p(y|x,\theta_0) \tag{A.14}$$

and

$$\begin{aligned}\mathcal{Q}(\theta,\theta_0) &= \mathcal{L}(p(y|x,\theta_0),\theta) && \text{(A.15)}\\ &= \int_{\mathcal{Y}} p(y|x,\theta_0)\log\frac{p(x,y|\theta)}{p(y|x,\theta_0)}\,\mathrm{d}y && \text{(A.16)}\\ &. && \text{(A.17)}\end{aligned}$$

If we rewrite the formula as

$$\begin{aligned}\mathcal{Q}(\theta,\theta_0) &= \int_{\mathcal{Y}} p(y|x,\theta_0)\log p(x,y|\theta)\,\mathrm{d}y - \int_{\mathcal{Y}} p(y|x,\theta_0)\log p(y|x,\theta_0)\,\mathrm{d}y && \text{(A.18)}\\ &= \mathrm{E}_{p(y|x,\theta_0)}\left[\log p(x,y|\theta)\right] + \text{const}\ , && \text{(A.19)}\end{aligned}$$

we see that the objective is to maximize the expectation of the joint log-probability given the old posterior of the hidden variables, hence the name "expectation maximization". Another useful form of the objective can be derived using (A.13):

$$\mathcal{Q}(\theta,\theta_0) = \mathrm{E}_{p(y|x,\theta_0)}\left[\log p(x|y,\theta)\right] - \mathrm{D}_{\mathrm{KL}}\left(p(y|x,\theta_0)\|p(y|\theta)\right)\ , \tag{A.20}$$

i.e. the auxilary function is sum of the expectation of the log-likelihood and negative divergence from the posterior distribution of the hidden variables to their prior.

The M-step

The M-step finds the new parameters θ^{new} by maximizing (A.15):

$$\theta^{\text{new}} = \arg\max_{\theta} \mathcal{Q}(\theta,\theta_0)\ . \tag{A.21}$$

Note that in the context of (A.20), θ parameterizes two distributions: (i) the likelihood of the observed variables $p(x|y,\theta)$, and (ii) the prior of the hidden variables $p(y|\theta)$. If we assume that the two distributions are parameterized by two disjoint subsets of parameters, i.e., $\theta = \langle\theta^{(x)},\theta^{(y)}\rangle$, we can rewrite (A.20) as

$$\mathcal{Q}(\theta,\theta_0) = \mathrm{E}_{p(y|x,\theta_0)}\left(\log p(x|y,\theta^{(x)})\right) - \mathrm{D}_{\mathrm{KL}}\left(p(y|x,\theta_0)\|p(y|\theta^{(y)})\right)\ , \tag{A.22}$$

and we can optimize the two sets of parameters independently. Optimizing the second term is referred to as *minimum divergence* and the two optimization terms are complementary. Note that minimum divergence is used, e.g. for updating the weights of the GMM (as they define the priors for the mixture components).

A.3 EM for PLDA

Let us give a quick cook-book style overview of how to exploit the EM to train the PLDA parameters. This section is mainly motivated by [Brümmer, 2010], where the complete derivations of the formulae can be found. The procedure is similar to training the parameters of the GMM subspace models. Again, we construct the objective function from the complete data log-likelihood and the prior using the same trick as in (3.45).

Let us recall that the observation i of speaker s is given as an i-vector $\boldsymbol{\phi}_{s,i}$ which can be decomposed using (5.24). As was mentioned earlier, we assume zero mean and we model $\boldsymbol{\phi}_{s,i}$ as

$$\boldsymbol{\phi}_{s,i} = \mathbf{V}\mathbf{y}_s + \mathbf{U}\mathbf{x}_{s,i} + \boldsymbol{\epsilon} \ . \tag{A.23}$$

The parameters of the model are $\theta = \langle \mathbf{V}, \mathbf{U}, \mathbf{D} \rangle$, where $\mathbf{D}$ is the precision matrix of the posterior distribution of $\boldsymbol{\epsilon}$ defined by (5.25).

A.3.1 Data

Let $\mathcal{S}$ be the set of all speakers, let $\boldsymbol{\Phi}_s$ be an $F \times n_s$ matrix of all n_s observations of speaker $s \in \mathcal{S}$

$$\boldsymbol{\Phi}_s = \begin{bmatrix} \boldsymbol{\phi}_{s,1} & \dots & \boldsymbol{\phi}_{s,n_s} \end{bmatrix} \ , \tag{A.24}$$

with F being the dimensionality of the i-vector, and let $\mathbf{X}_s$ be the matrix of all hidden variables $x_{s,i}$

$$\mathbf{X}_s = \begin{bmatrix} \mathbf{x}_{s,1} & \dots & \mathbf{x}_{s,n_s} \end{bmatrix} \ . \tag{A.25}$$

Let us define the sufficient statistics in a similar way as we did for the GMM. The zero-order statistics for speaker s are directly given by n_s, the global zero-order statistics are given as

$$N = \sum_{s \in \mathcal{S}} n_s \ , \tag{A.26}$$

the first-order statistiscs for speaker s are given as

$$\mathbf{f}_s = \sum_{i=1}^{n_s} \boldsymbol{\phi}_{s,i} \ , \tag{A.27}$$

and the second-order statistics for all observations are given as

$$\mathbf{S} = \sum_{s \in \mathcal{S}} \sum_{i=1}^{n_s} \boldsymbol{\phi}_{s,i} \boldsymbol{\phi}'_{s,i} \ . \tag{A.28}$$

A.3.2 The Log-Likelihood Function

The complete data log-likelihood for speaker s is given as

$$\log p(\boldsymbol{\Phi}_s | \mathbf{y}_s, \mathbf{X}_s, \theta) = \sum_{i=1}^{n_s} \log \mathcal{N}\left(\boldsymbol{\phi}_{s,i} | \mathbf{V}\mathbf{y}_s + \mathbf{U}\mathbf{x}_{s,i}, \mathbf{D}^{-1}\right) \tag{A.29}$$

$$\propto \sum_{i=1}^{n_s} \left(-\frac{1}{2} \boldsymbol{\phi}'_{s,i} \mathbf{D} \boldsymbol{\phi}_{s,i} + \boldsymbol{\phi}'_{s,i} \mathbf{D} \mathbf{V} \mathbf{y}_s + \boldsymbol{\phi}'_{s,i} \mathbf{D} \mathbf{U} \mathbf{x}_{s,i} \right.$$
$$\left. - \frac{1}{2} \mathbf{y}'_s \mathbf{V}' \mathbf{D} \mathbf{V} \mathbf{y}_s - \mathbf{y}'_s \mathbf{V}' \mathbf{D} \mathbf{U} \mathbf{x}_{s,i} - \frac{1}{2} \mathbf{x}'_{s,i} \mathbf{U}' \mathbf{D} \mathbf{U} \mathbf{x}_{s,i} \right) \ . \tag{A.30}$$

A.3.3 Hidden-Variable Prior

The joint prior for the hidden variables for speaker s is given as

$$p(\mathbf{X}_s, \mathbf{y}_s) = p(\mathbf{X}_s)p(\mathbf{y}_s) \tag{A.31}$$

$$\propto \exp\left(-\frac{1}{2}\mathbf{y}_s'\mathbf{y}_s - \frac{1}{2}\operatorname{tr}(\mathbf{X}_s'\mathbf{X}_s)\right) \tag{A.32}$$

A.3.4 Hidden-Variable Posterior

For convenience, let us define these substitutions:

$$\mathbf{J} = \mathbf{U}'\mathbf{D}\mathbf{V} \tag{A.33}$$

$$\mathbf{K} = \mathbf{U}'\mathbf{D}\mathbf{U} + \mathbf{I} \tag{A.34}$$

$$\mathbf{L}_s = n_s\mathbf{V}'\mathbf{D}\mathbf{V} + \mathbf{I}\ . \tag{A.35}$$

The posterior of the hidden variables is given as a joint probability

$$p(\mathbf{X}_s, \mathbf{y}_s|\mathbf{\Phi}_s, \theta) = p(\mathbf{X}_s|\mathbf{y}_s, \mathbf{\Phi}_s, \theta)p(\mathbf{y}_s|\mathbf{\Phi}_s, \theta)\ , \tag{A.36}$$

with the individual posteriors

$$p(\mathbf{X}_s|, \mathbf{y}_s, \mathbf{\Phi}_s, \theta) = \prod_{i=1}^{n_s} \mathcal{N}\left(\mathbf{x}_{s,i}; \hat{\mathbf{x}}_{s,i}, \mathbf{K}^{-1}\right) \tag{A.37}$$

$$p(\mathbf{y}_s|\mathbf{\Phi}_s, \theta) = \mathcal{N}\left(\mathbf{y}_s|\hat{\mathbf{y}}_s, \mathbf{P}_s^{-1}\right)\ , \tag{A.38}$$

where

$$\mathbf{P}_s = n_s\left(\mathbf{V}'\mathbf{D}\mathbf{V} - \mathbf{J}'\mathbf{K}^{-1}\mathbf{J}\right) + \mathbf{I} \tag{A.39}$$

$$\hat{\mathbf{x}}_{s,i} = \tilde{\mathbf{x}}_{s,i} - \mathbf{K}^{-1}\mathbf{J}\mathbf{y}_s \tag{A.40}$$

$$\hat{\mathbf{y}}_s = \mathbf{P}_s^{-1}\left(\mathbf{V}'\mathbf{D}\mathbf{f}_s - \sum_{i=1}^{n_s}\mathbf{J}'\tilde{\mathbf{x}}_{s,i}\right) \tag{A.41}$$

with

$$\tilde{\mathbf{x}}_{s,i} = \mathbf{K}^{-1}\mathbf{U}'\mathbf{D}\boldsymbol{\phi}_{s,i}\ . \tag{A.42}$$

A.3.5 Marginal Log-Likelihood (EM Objective)

As in the GMM subspace modeling, the EM objective is defined as a marginal log-likelihood over the hidden variables. Again, we could marginalize by integrating over the hidden variables, but we can use the same approach as in the GMM subspace modeling:

$$p(\mathbf{\Phi}_s|\theta) = \left.\frac{p(\mathbf{\Phi}_s|\mathbf{X}_s, \mathbf{y}_s, \theta)p(\mathbf{X}_s)p(\mathbf{y}_s)}{p(\mathbf{X}_s|\mathbf{\Phi}_s, \mathbf{y}_s, \theta)p(\mathbf{y}_s|\mathbf{\Phi}_s, \theta)}\right|_{\mathbf{y}_s=\mathbf{0},\mathbf{X}_s=\mathbf{0}}\ . \tag{A.43}$$

The EM objective is given as a log of (A.43), summed over all speakers:

$$\mathcal{L}(\theta) = \sum_{s\in\mathcal{S}} \log p(\mathbf{\Phi}_s|\theta) \tag{A.44}$$

$$= \sum_{s\in\mathcal{S}} \sum_{i=1}^{n_s} -\frac{1}{2}\boldsymbol{\phi}'_{s,i}\mathbf{D}\boldsymbol{\phi}_{s,i} - \frac{1}{2}\log|\mathbf{P}_s| + \frac{1}{2}\hat{\mathbf{y}}'_s\mathbf{P}_s\hat{\mathbf{y}}_s + \frac{1}{2}\hat{\mathbf{x}}'_{s,i}\mathbf{K}\hat{\mathbf{x}}_{s,i}\ . \tag{A.45}$$

A.3.6 E-step

Let us stack the hyper parameters and the hidden variables as

$$\mathbf{W} = [\mathbf{U}\ \mathbf{V}] \tag{A.46}$$

$$\mathbf{z}_{s,i} = \begin{bmatrix} \mathbf{x}_{s,i} \\ \mathbf{y}_s \end{bmatrix}\ . \tag{A.47}$$

The auxillary function is defined as

$$\mathcal{Q}(\theta,\theta_0) = \mathrm{E}_{\theta_0}\left[\sum_{s\in\mathcal{S}} \log p\left(\mathbf{\Phi}_s|\mathbf{y}_s,\mathbf{X}_s,\theta\right) + \mathrm{const}\right] \tag{A.48}$$

$$= \frac{N}{2}\log|\mathbf{D}| - \frac{1}{2}\operatorname{tr}(\mathbf{SD}) - \frac{1}{2}\operatorname{tr}(\mathbf{RW}'\mathbf{DW}) + \operatorname{tr}(\mathbf{TDW})\ , \tag{A.49}$$

where $\mathbf{S}$ is the matrix of the global second-order statistics, and the accumulators $\mathbf{R}$ and $\mathbf{T}$ are computed using the expected values of $\mathbf{z}_s$ via the posterior computed using θ_0:

$$\mathbf{T} = \sum_{s\in\mathcal{S}} \sum_{i=1}^{n_s} \mathrm{E}\left[\mathbf{z}_{s,i}\right] \boldsymbol{\phi}'_{s,i} \tag{A.50}$$

$$= \sum_{s\in\mathcal{S}} \sum_{i=1}^{n_s} \begin{bmatrix} \mathbf{T}_{\mathbf{x}} \\ \mathbf{T}_{\mathbf{y}} \end{bmatrix} \tag{A.51}$$

and

$$\mathbf{R} = \sum_{s\in\mathcal{S}} \sum_{i=1}^{n_s} \mathrm{E}\left[\mathbf{z}_{s,i}\mathbf{z}'_{s,i}\right] \tag{A.52}$$

$$= \sum_{s\in\mathcal{S}} \sum_{i=1}^{n_s} \begin{bmatrix} \mathbf{R}_{\mathbf{xx}} & \mathbf{R}_{\mathbf{xy}} \\ \mathbf{R}'_{\mathbf{xy}} & \mathbf{R}_{\mathbf{yy}} \end{bmatrix}, \tag{A.53}$$

where

$$\mathbf{T}_\mathbf{y} = \sum_{s\in\mathcal{S}} \sum_{i=1}^{n_s} \hat{\mathbf{y}}_s \boldsymbol{\phi}'_{s,i} = \sum_{s\in\mathcal{S}} \hat{\mathbf{y}}_s \mathbf{f}'_s \tag{A.54}$$

$$\mathbf{T}_\mathbf{x} = \sum_{s\in\mathcal{S}} \sum_{i=1}^{n_s} \mathrm{E}\left[\hat{\mathbf{x}}_{s,i}(\mathbf{y})'\right] \boldsymbol{\phi}'_{s,i} \tag{A.55}$$

$$= \sum_{s\in\mathcal{S}} \sum_{i=1}^{n_s} \mathbf{K}^{-1}\left(\mathbf{U}'\mathbf{D}\boldsymbol{\phi}_{s,i} - \mathbf{J}\hat{\mathbf{y}}_s\right) \boldsymbol{\phi}'_{s,i} \tag{A.56}$$

$$= \mathbf{K}^{-1}\left(\mathbf{U}'\mathbf{D}\mathbf{S} - \mathbf{J}\mathbf{T}_\mathbf{y}\right) \tag{A.57}$$

and

$$\mathbf{R}_{\mathbf{yy}} = \sum_{s\in\mathcal{S}} \sum_{i=1}^{n_s} \mathrm{E}\left[\mathbf{y}_s\mathbf{y}'_s\right] = \sum_{s\in\mathcal{S}} n_s\left(\mathbf{P}_s^{-1} + \hat{\mathbf{y}}_s\hat{\mathbf{y}}'_s\right) \tag{A.58}$$

$$\mathbf{R}_{\mathbf{xy}} = \sum_{s\in\mathcal{S}} \sum_{i=1}^{n_s} \mathrm{E}\left[\mathbf{x}_{s,i}\mathbf{y}'_s\right] = \mathbf{K}^{-1}\left(\mathbf{U}'\mathbf{D}\mathbf{T}_\mathbf{y} - \mathbf{J}\mathbf{R}_{\mathbf{yy}}\right) \tag{A.59}$$

$$\mathbf{R}_{\mathbf{xx}} = \sum_{s\in\mathcal{S}} \sum_{i=1}^{n_s} \mathrm{E}\left[\mathbf{x}_{s,i}\mathbf{x}'_{s,i}\right] \tag{A.60}$$

$$= \mathbf{K}^{-1}\left(\mathbf{U}'\mathbf{D}\mathbf{S}\mathbf{D}\mathbf{U} - \mathbf{U}'\mathbf{D}\mathbf{T}'_\mathbf{y}\mathbf{J}' - \mathbf{J}\mathbf{T}_\mathbf{y}\mathbf{D}\mathbf{U} + \mathbf{J}\mathbf{R}_{\mathbf{yy}}\mathbf{J}'\right)\mathbf{K}^{-1} \tag{A.61}$$

A.3.7 M-step

Differentiating (A.49) w.r.t. **W** and setting to zero gives

$$\mathbf{W}_{\mathrm{em}} = \mathbf{T}'\mathbf{R}^{-1}\,. \tag{A.62}$$

Differentiating w.r.t. **D** gives

$$\mathbf{D}_{\mathrm{em}} = N\left(\mathbf{S} - \mathbf{T}'\mathbf{R}^{-1}\mathbf{T}\right)^{-1}\,, \tag{A.63}$$

and if we want to force **D** to be isotropic, i.e. $\mathbf{D} = d\mathbf{I}$, then

$$d = \frac{ND}{\mathrm{tr}\left(\mathbf{S} - \mathbf{T}'\mathbf{R}^{-1}\mathbf{T}\right)}\,. \tag{A.64}$$

The MD step is again based on rotating the space to compensate for the updated priors as:

$$\mathbf{U}_{\mathrm{md}} = \mathbf{U}_{\mathrm{em}}\,\mathrm{Chol}(\mathbf{U}_{\mathrm{rot}})' \tag{A.65}$$

$$\mathbf{V}_{\mathrm{md}} = \mathbf{V}_{\mathrm{em}}\,\mathrm{Chol}(\mathbf{V}_{\mathrm{rot}})' + \mathbf{U}_{\mathrm{md}}\mathbf{G}\,, \tag{A.66}$$

where

$$\mathbf{G}' = \mathbf{R}_{\mathbf{yy}}^{-1}\mathbf{R}'_{\mathbf{xy}} \tag{A.67}$$

$$\mathbf{V}_{\mathrm{rot}} = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} \mathbf{P}_s^{-1} + \hat{\mathbf{y}}_s\hat{\mathbf{y}}'_s \tag{A.68}$$

$$\mathbf{U}_{\mathrm{rot}} = \frac{1}{N}\left(\mathbf{R}_{\mathbf{xx}} - \mathbf{G}\mathbf{R}'_{\mathbf{xy}}\right) \ , \tag{A.69}$$

and Chol(·) denotes Cholesky decomposition, where $\mathrm{Chol}(\mathbf{A})\,\mathrm{Chol}(\mathbf{A})' = \mathbf{A}$.