Západočeská Univerzita v Plzni
Fakulta Aplikovaných Věd

Vysokodimenzionální Prostory a Modelování v úloze Rozpoznávání Řečníka

Ing. Lukáš Machlica

DISERTAČNÍ PRÁCE
k získání akademického titulu doktor
v oboru **Kybernetika**

Školitel: Doc. Dr. Ing. Vlasta Radová
Katedra Kybernetiky

Plzeň, 2012

University of West Bohemia
Faculty of Applied Sciences

High Dimensional Spaces and Modelling in the task of Speaker Recognition

Ing. Lukáš Machlica

DOCTORAL THESIS

submitted in partial fulfilment of the requirements
for the degree Doctor of Philosophy
in the field of

Cybernetics

Advisor: Doc. Dr. Ing. Vlasta Radová
Department of Cybernetics

Pilsen, 2012

Acknowledgements

My big thanks belong to all who have encouraged, helped and supported me during my studies, especially to my parents and my family. Thank you!

I would like to thank my advisor Doc. Dr. Ing. Vlasta Radová and also Doc. Ing. Luděk Müller Ph.D. for advices they gave me.

For the financial assistance I would like to thank the Ministry of Education, Youth and Sport of the Czech Republic under the project MŠMT LC536 (Integrated Center for Natural Language Processing), and for the financial assistance of the Grant Agency of the Czech Republic project No. GAČR P103/12/G084 (Center for Large Scale Multi-modal Data Interpretation).

Abstract

The automatic speaker recognition made a significant progress in the last two decades. Huge speech corpora containing thousands of speakers recorded on several channels are at hand, and methods utilizing as much information as possible were developed. Nowadays state-of-the-art methods are based on Gaussian mixture models used to estimate relevant statistics from feature vectors extracted from the speech of a speaker, which are further concatenated into a high dimensional vector – supervector.

Methods concerning the extraction of high dimensional supervectors along with techniques capable to build a speaker model in such a high dimensional space are described in depth and links between these methods are found. The main emphasize is laid on the analysis of these methods and an efficient implementation in order to process huge amounts of development data to train the speaker recognition system. Also the influence of development corpora on the recognition performance is experimentally tested.

Keywords: Gaussian mixture models, support vector machine, supervector, factor analysis, dimensionality reduction, speaker recognition

Abstrakt

Během posledních dvou desetiletí bylo v úloze automatického rozpoznávání řečníka dosaženo výrazných pokroků. Byly nahrány obrovské řečové databáze obsahující tisíce řečníků mluvících na různých akustických kanálech. Zároveň byly vyvinuty metody, které se snaží z těchto dat extrahovat co nejvíce informací. Nejmodernější metody jsou založeny na modelech Gaussovských směsí. S jejich pomocí jsou z příznakových vektorů, extrahovaných z řečových dat řečníků, počítány statistiky. Tyto statistiky jsou následně zřetězeny/pospojovány do vysokorozměrných vektorů – supervektorů.

Práce se zabývá podrobným popisem metod extrakce vysokodimenzionálních supervektorů společně s technikami jejich modelování. Hlavní důraz je kladen na analýzu těchto metod, jejich propojení, a protože je při trénování systému rozpoznávání řečníka potřeba zpracovat velké množství vstupních dat, i na jejich efektivní implementaci. Experimentálně je také vyšetřen vliv dat pro trénování na kvalitu rozpoznávání.

Klíčová slova: model Gaussovských směsí, support vector machine, supervektor, faktorová analýza, redukce dimenze, rozpoznávání mluvčích

Prohlášení

Prohlašuji, že jsem tuto disertační práci vypracoval samostatně, s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne Lukáš Machlica

Contents

List of Abbreviations

BDK	Basic Derivative Kernel
BN	Bayesian Network
BOC	Bayes Optimal Classifier
CDHMM	Continuous Density Hidden Markov Model
CMLLR	Constrained Maximum Likelihood Linear Regression
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DCF	Decision Cost Function
DET	Detection Error Trade-off
EER	Equal Error Rate
EM	Expectation Maximization
FA	Factor Analysis
FC	Fusion Coefficient
FI	Fisher Information
FLOPS	FLoating-point OPerations Per Second
fMLLR	feature Maximum Likelihood Linear Regression
FSH	Fisher English Training Speech Part 1 and Part 2
FSW	Feature Space Whitening
FW	Feature Warping
GDK	Generalized Derivative Kernel
GFLOPS	Giga FLOPS
GLDS	Generalized Linear Discriminant Sequence
GM	Global memory
GMM	Gaussian Mixture Model

GN	Gaussian Normalization
GP	Gaussian Process
GPLVM	Gaussian Process Latent Variable Model
GPU	Graphics Processing Unit
GSV	Gaussian-mean Supervector
HMM	Hidden Markov Model
JFA	Joint Factor Analysis
KKT	Karush-Kuhn-Tucker
KLD	Kulback-Leibler Divergence
L^2IPK	L^2 Inner Product Kernel
LDA	Linear Discriminant Analysis
LFCC	Linear Frequency Cepstral Coefficients
LLR	Log-Likelihood Ratio
LLRS	Log-Likelihood Ratio Score
LLS	Log-Likelihood Score
LR	Logistic Regression
LS	Least Squares
LVCSR	Large-Vocabulary Continuous Speech Recognition system
MAB	Memory-Aligned Block
MAP	Maximum A-posteriory Probability
MD	Minimum Divergence
MFCC	Mel Frequency Cepstral Coefficients
minDCF	minimum of the Decision Cost Function
ML	Maximum Likelihood
MLE	Maximum Likelihood Estimation
MLLR	Maximum Likelihood Linear Regression
MRF	Markov Random Field
NAP	Nuisance Attribute Projection
NN	Neural Networks
OCK	One-Class Kernel

OCKD	One-Class Kernel Diagonal
PCA	Principal Component Analysis
PDK	Probabilistic Distance Kernel
PLDA	Probabilistic Linear Discriminant Analysis
PPCA	Probabilistic Principal Component Analysis
PS	Probability Sequence
RBF	Radial Basis Function
RN	Rank Normalization
RT	Regression Tree
SD	Speaker Dependent
SI	Speaker Independent
SIMD	Single Instruction, Multiple Data
SKLR	Sparse Kernel Logistic Regression
SLK	Supervector Linear Kernel
SM	Shared Memory
SN	Spherical Normalization
SNR	Signal-to-Noise Ratio
SPFP	Single-Precision Floating Point
SR	Speaker Recognition
SRE	Speaker Recognition Evaluation
SSE	Streaming SIMD Extension
SV	SuperVector
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SWC	Switchboard Cellular Audio Part 1 and Part 2
T-norm	Test Normalization
TM	Texture Memory
UBM	Universal Background Model
VAD	Voice Activity Detector
VT	Vocal Tract

WCCN	Within Class Covariance Normalization
Z-norm	Zero Normalization
ZT-norm	Zero and Test Normalization

List of Functions

diag($\boldsymbol{X}$)	transforms the matrix $\boldsymbol{X}$ to a vector taking only the diagonal of the input matrix $\boldsymbol{X}$
diagz($\boldsymbol{X}$)	zeros all the non-diagonal elements of the matrix $\boldsymbol{X}$
DIAG($\boldsymbol{x}$)	creates a diagonal matrix with entries of $\boldsymbol{x}$ on its diagonal

List of Figures

List of Tables

Chapter 1

Introduction

In the 20th century a massive boom of technological progress has begun. With the arrival of computers more and more tasks, involving complicated mathematical approaches, become solvable. Mathematical algorithms have been spread and successfully applied to many scientific branches, among others, to Speaker Recognition (SR). The domain of SR was until then fully in the hands of phoneticians and the main interest of SR laid in the area of forensics. However, it has been shown that SR may be applied to many other newly formed tasks. E.g. in the task of speech recognition the speaker's identity can be utilized to adjust a speaker independent model to better fit the voice of the talking person. Another usage of SR can be found in problems where huge databases of spoken speech are searched through and only utterances originating from one specific speaker/source are requested (e.g. in telecommunications or in security domains).

The task of speaker recognition is usually divided into problems of *identification* and *verification*. In the verification case a decision has to be made, whether the speaker really is who he claims to be. Thus, it is a one-to-one comparison. In the case of identification a set of reference speakers with known identities is given. The task can be further divided into *closed set identification* and *open set identification*. If the closed set identification is considered, we assume that the unknown speaker is a speaker from the set of reference speakers. In the open set identification such an assumption is broken[1]. The problem of SR can be distinguished also upon the dependence on the text content inherent in the speech. The *text-dependent* SR focuses on special phonetic events (e.g. vowels, syllables, phrases or words) present in the spoken speech, whereas the *text-independent* speaker recognition puts no limitation on what has been said.

Generally, the speech sound is a product of air expiration from lungs and of a Vocal Tract (VT) configuration. An important assumption concerning biometrics (e.g. the usage of SR in forensics) is that the VT and its configuration when a speech sound is produced changes between speakers, and in fact it is assumed that these characteristics are unique. Hence, the task of SR may be stated as an effort to capture specific shapes of VT through investigation of the acoustic sound wave produced by the speaker.

The problem of automatic SR can be divided into several phases:

1. *Feature extraction* – the varying time sequence of samples (amplitudes of recorded speech waves)

[1] Note that in all the cases a one-to-one comparison is performed and a verification score is computed given recordings of two speakers (the unknown and the reference speaker). The verification score is further processed according to the stated task: *closed set identification* – the unknown speaker is the reference speaker with the highest verification score; *open set identification* – after the most similar speaker from the reference set of speakers is found, the verification score is compared to a verification threshold, if the score is higher than the threshold the identity is verified, denied otherwise. Therefore the terms speaker recognition and speaker verification will be often interchanged, and in the context of this thesis no difference is made between them.

is processed to gain information regarding the speaker's identity, and feature vectors are extracted. Feature vectors lie in a feature space – more precisely in speaker specific feature subspaces.

2. *Modelling stage* – a model is build in order to capture relevant dependencies/properties in the feature set of each speaker.

3. *Recognition phase* – given a speaker (target) model and a set of feature vectors of an unknown (test) speaker a recognition score is computed and further handled according to the stated task (verification/identification).

However, nowadays the border between individual phases is not clear; i.e. in the feature extraction process also models can be trained in order to extract higher level features (will be described soon), instead of working with feature vectors models can be extracted also for test speakers and in the recognition phase models are compared. Moreover, all of these phases often incorporate variety of normalization techniques, which will be described in Section 5.

The feature extraction process can be regarded as the first and the most important step for further manipulation with the data, which can be:

1. *Classification* – a set of classes is given and each feature has to be assigned to one of them. Thus, it would be wise to choose features in such a way that the between-class variation would be as high as possible and the within-class variation would be as low as possible.

2. *Representation* – we want to extract features that capture the true nature of the data; we may not care whether extracted information is shared in greater extent between all classes/speakers, and therefore does not contain any discriminative information (is not suitable for classification).

Both tasks significantly overlap and they both assume and should also reflect the understanding of behaviour, structure, correctness of the data, etc. Complications like channel distortions, data corruption, noise presence and others also have to be considered. It is a very common practice to impose some presumptions (e.g. on independence or distribution of samples) to facilitate the solving of the problem. Thus, the true nature of the data can be distorted and some inaccuracies may come forth. Obviously, the feature extraction process has to be carried out with caution and should be preceded by succession of experiments, which would reveal the mentioned characteristics of the data. Lot of feature extraction techniques concerning the spoken speech have been developed through time. The aim of this thesis is the modelling of features in the feature space, therefore the reader is referred to works [1, 2, 3] concerning the process of feature extraction from an acoustic sound wave.

Once feature vectors have been extracted, a model is estimated for each speaker. For more than a decade Gaussian Mixture Models (GMMs) dominated the task of SR. They belong to the class of generative models since samples can be drawn given an estimate of model parameters. Another example of a generative model is Hidden Markov Model (HMM), which in addition captures the dependence of successive feature vectors. These methods will be discussed in Chapter 2. Nowadays, GMMs play still an important role in the state-of-the-art speaker recognition systems, however they are used mainly to delimit and split up the feature space according to level of interest, and to extract data statistics related to distinct parts of the feature space. This is done via estimation of an Universal Background Model (UBM) comprising many GMM components and trained on a huge amount of development data. All the acoustic conditions in which the system will be used should be covered. Loosely speaking, UBM should model the acoustic environment of a specific SR task. Subsequently, given an UBM, statistics of extracted feature vectors of a speaker, related to distinct parts of the feature space (i.e. to individual Gaussians in the UBM), are estimated. Next, a supervector (SV)

is formed by concatenation of these statistics in accordance with GMM components in the UBM, yielding the SV of substantially high dimension.

For example, given a set of feature vectors of a speaker and a trained UBM, Maximum A-Posteriory (MAP) adaptation discussed in Chapter 3 is used to estimate a GMM of this speaker. In order to get a SV, means of the MAP adapted GMM are concatenated. The MAP adaptation allows that parts of the UBM that represent only the acoustic environment remain unchanged for all the speakers. Moreover, in the adaptation process we are able to track changes of each UBM Gaussian (mixture component), more precisely we are able to determine, in which direction and how far has been each of the mixture components moved to fit the speaker's data. Extraction of several types of SVs is explained in Chapter 4.

Note that the extraction of SVs can be seen as a higher level feature extraction, where already an estimation phase of a generative model (UBM) was incorporated.

Simultaneously two techniques to handle the high dimensional SVs were proposed. The first is based on Support Vector Machines (SVMs) as a discriminative trainer described in Section 2.3.1, which have very good generalization properties and are well suited for the task of modelling when only a few (in the case of SVs often only one) examples/supervectors of a speaker/class are available. Since both generative (UBM/GMM) and discriminative (SVM) modelling are utilized, the techniques comprising GMM based SVs and SVMs are also known as *hybrid modelling* [4]. The concept of SVs and SVM was further extended by the Nuisance Attribute Projection (NAP) addressed in Section 5.7, which is used to suppress undesirable channel variabilities between sessions (recordings on distinct channels) of one speaker. NAP is based on an orthogonal projection, where directions most vulnerable to environment/channel changes are projected out.

The latter technique (more precisely, a set of techniques) is based on Factor Analysis (FA) and generative modelling, and it is discussed in depth in Chapter 6. The idea is that since the dimensionality of SVs is in comparison with the number of development speakers very high, many dimensions have to be correlated with each other. Hence, the effective information on the identity of speakers has to lie in a much lower subspace. Moreover, since several sessions of one speaker are available, one could determine not only the speaker identity subspace, but also the channel/session subspace, which should be also of a much lower dimension. These principles were incorporated into a method called Joint Factor Analysis (JFA), where the word *joint* refers to the fact that not only the speaker, but also the channel variabilities are treated in one JFA model. However, experiments in [5] have shown, that the channel/session subspace does still contain some substantial information concerning the identity of a speaker. Therefore, JFA was extended to the concept of i-vectors discussed in Section 6.4. The main difference between JFA and i-vectors is that i-vectors do not distinguish between the speaker and the channel space. They work with a *total variability space* containing simultaneously speaker and channel variabilities, whereas JFA treats both spaces individually. Details on both techniques can be found in Chapter 6.

Independently of JFA a method called Probabilistic Linear Discriminant Analysis (PLDA), described in Section 6.2, has been developed in the computer vision to tackle the problem of face recognition. PLDA is very similar to JFA, it decomposes the feature space to speaker and channel dependent subspaces, but rather than GMM based SVs ordinary feature vectors are utilized. The difference between GMM based SVs treated in JFA/i-vectors and ordinary vectors is that distinct dimensions of GMM based SVs are weighted when estimating the subspace decomposition, details are given in Section 6.3.1. Since PLDA is a generative model, it allows to compute the probability that several i-vectors originate from the same source, and thus it is well suited as a verification tool for a speaker recognition system [6].

The system examined in this thesis will use SVs, SVMs, i-vectors and PLDA models along with distinct normalization techniques.

1.1 Aim of the Thesis and the Novelties

Generally, the thesis is devoted to the problem of modelling of feature sets for classification in situations where lots of data are available, but the work will be strongly oriented toward the task of open set, text independent speaker identification.

The crucial problem when implementing a state-of-the-art SR system composed of modules such as JFA, SVM, i-vector extractor or PLDA is that huge amount of development data from a lot of speakers are required, moreover several sessions have to be available for each speaker in order to train a reliable model. Therefore three main problems are faced in this thesis: *acceleration of algorithms*, *influence of development data* on the performance of the SR system, and *connections between presented methods*. However, the goal of the work is also to *describe, explain and understand the principles of individual modelling techniques* in a wider context.

In relation to the main goals mentioned above following novel approaches are presented in this work:

1. As mentioned earlier, when preparing a SR system at first UBM has to be trained from a huge amount of development data (several thousands of hours, see Section 7.1). The estimation process is based on Expectation Maximization (EM) algorithm based on maximization of the data likelihood given the model. Moreover, the data statistics from the EM algorithm are utilized also in adaptation algorithms (e.g. MAP adaptation), which are used to extract SVs. Hence, it is in great demand to have a really fast and robust implementation. For this purpose parallel technologies like supercomputers, clusters, grids, and cloud infrastructures may be utilized. However, in order to fully exploit the potential of the computing power, the algorithm has to be parallelized in a proper way and the memory management has to be handled too. Since supercomputers are not easily and broadly available, we have focused on the computing power provided by the Graphics Processing Unit (GPU), which developed through time to a highly parallel and computationally powerful tool for high performance computing [7]. GPUs are widespread and easily available for a reasonable price. For all this reasons in Chapter 8 a highly efficient parallel implementation on a GPU is proposed leading to a *hundreds times faster* EM algorithm and statistics extraction for UBM, GMM and SV estimation. Since this chapter contains the (GPU) implementation of an estimation algorithm without any theoretical background, but the results are of high significance, the last chapter is devoted to its description rather than a section in the appendix.

2. Next novel approach concerns the PLDA model estimation, where each speaker/individual has to contain several examples/sessions. The one described in [8] is extremely slow mainly for high dimensional feature vectors containing distinct numbers of examples for each individual. In addition, the operating principle of the method stays hidden. Utilizing theorems from linear algebra concerning the inversion of matrices along with some suitable rearrangements of formulas and noticing the stand-alone summation terms a much more efficient training algorithm is proposed in Section 6.2.2 and Section 6.2.3. In addition, the background of the algorithm is clarified. The revisited algorithm is thousands time faster than a naive implementation assuming a large dataset to be processed, see Section 7.7.3.

3. Two kinds of methods dominate the task of speaker recognition: based on eigenvector decomposition such as Principal Component Analysis (PCA) and Nuisance Attribute Projection (NAP, described in Section 5.7), and based on Factor Analysis (FA). Therefore a special section is devoted to the analysis of their similarities and dissimilarities. Formulations of both methods are

converted to the problem of Least Squares (LS), and in the light of LS they are simultaneously analysed. All the details are given in Section 6.5.

4. In Chapter 7 experiments on state-of-the-art SR systems are performed utilizing data from NIST Speaker Recognition Evaluations (SREs) 2008 and 2010. It is shown in what extent do additional normalization techniques help to decrease the error rates, results are analysed, and also the dimensionality reduction of SVM models is examined in Section 7.6. Since SVM is used to train a speaker model associated with a speaker dependent SV, kernels (used to map – in a linear or non-linear way – input vectors to some other favourable feature space) proposed for SVs related to UBM are tested and the results are inspected.

5. Since huge amounts of development data are available, the question is how does a system behave when the amount of development data changes. The focus is laid on the system based on i-vectors and a PLDA model. Several development subsets are created and one PLDA model is trained for each of them. Moreover, two alternatives are experimentally tested: pooling all the development data and training one PLDA model, or training one PLDA model for each development subset and fusing the verification results. Experiments in Section 7.7 are provided with analysis of the results.

6. Finally, the complementarity of implemented methods is examined in Section 7.8. Since a variety of methods was tested based on generative and discriminative modelling, subspace decomposition, dimensionality reduction, etc. the combination of these methods/systems should bring additional improvements to the recognition unless one of the techniques significantly outperforms the rest of the methods. Experiments are provided with the discussion on the differences between tested methods.

Chapter 2

Classifiers

The classification can be seen as a mapping of feature space vectors into a finite set of labels (classes/clusters). The basic assumption is that the vectors does form clusters. In the speaker recognition such an assumption is met, however it may be that the clusters cannot be recognized. Each speaker s can be regarded as a separate class, and the set of (parametrized) speaker data $\boldsymbol{X}_s$ as the set of respective feature vectors. The task of a classifier (decision function) $\mathcal{D}(\boldsymbol{x})$ is to decide on the pertinence of an input data to one of the classes (speakers), hence label the input data in agreement with a set of reference labels y_s assigned to reference speakers. For the sake of clarity let's state, that reference speakers are those speakers whose data were seen during the training of a classifier. In this chapter we will assume (without the loss of generality) that each portion of data corresponds to one of the given classes (e.g. each speaker's data were seen during the training). The classifier can be expressed as a mapping in the form

$$\mathcal{D} : \boldsymbol{X} \mapsto \boldsymbol{Y}, \quad \boldsymbol{Y} = \{y_1, \ldots, y_Q\}, \quad y_q = \mathcal{D}(\boldsymbol{x}) \,, \tag{2.1}$$

where $\boldsymbol{Y}$ is the finite set of Q reference labels, and $\boldsymbol{X}$ represents a set of feature vectors spread in the feature space according to a probability density function $p(\boldsymbol{X})$. In the following, the concept of an Bayes Optimal Classifier (BOC) will be discussed, which can be thought of as a theoretical base of the classification task. Subsequently the parametric and nonparametric algorithms will be described and the main emphasis will be laid on parametric statistical models (e.g. Gaussian Mixture Models) and nonparametric linear discriminants (e.g. Support Vector Machines).

2.1 Optimal Classifier

From the theoretical point of view an optimal classifier should minimize the *overall risk* R given as [9]

$$R(\mathcal{D}) = \int_{\boldsymbol{X}} R(\mathcal{D}(\boldsymbol{x})|\boldsymbol{x}) p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \tag{2.2}$$

where $R(\mathcal{D}(\boldsymbol{x})|\boldsymbol{x})$ represents the *conditional risk*,

$$R(\mathcal{D}(\boldsymbol{x})|\boldsymbol{x}) = \sum_{q=1}^{Q} l(\mathcal{D}(\boldsymbol{x})|y_q) p(y_q|\boldsymbol{x}) \,, \tag{2.3}$$

where $l(y_i|y_s)$ is the loss that will be taken when the feature vector $\boldsymbol{x}$ belonging to the class y_s will be labeled as y_i, hence classified into the i^{th} class. The loss can be regarded also as a cost function $c(\boldsymbol{x}, y_s, \mathcal{D}(\boldsymbol{x}))$ for $\mathcal{D}(\boldsymbol{x}) = y_i$, penalizing misclassifications so that the following holds

$$c(\cdot, \cdot, \cdot) \geq 0 \text{ and } c(\cdot, y_s, y_s) = 0 \,. \tag{2.4}$$

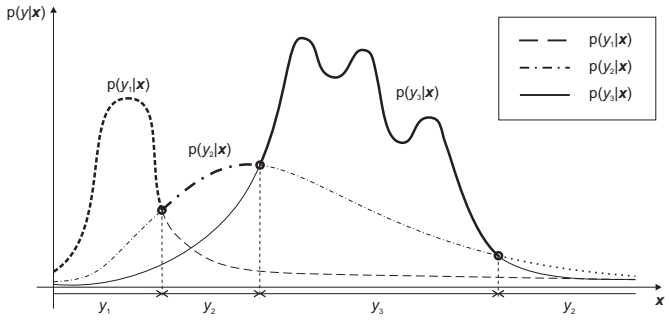

Figure 2.1: The a-posteriori probability distributions for classes y_1, y_2 and y_3 given the feature vector $\boldsymbol{x}$. The bold lines represent the optimal decision rule that should the optimal classifier $\mathcal{D}^*$ obey.

Now, the optimal classifier $\mathcal{D}^*$ should be chosen in order to minimize the following form

$$R(\mathcal{D}) = \sum_{q=1}^{Q} \int_{\boldsymbol{X}} c(\boldsymbol{x}, y_q, \mathcal{D}(\boldsymbol{x})) p(\boldsymbol{x}, y_q) \mathrm{d}\boldsymbol{x} \,. \tag{2.5}$$

Assuming $\boldsymbol{Y}$ continuous, open set of labels, the preceding equation can be rewritten as

$$R(\mathcal{D}) = \int_{\boldsymbol{X}\times\boldsymbol{Y}} c(\boldsymbol{x}, y, \mathcal{D}(\boldsymbol{x})) \mathrm{d}P(\boldsymbol{x}, y) \,. \tag{2.6}$$

Thus, all possible pairs of feature vectors and labels are considered and rated while (2.4) holds. One of the popular cost functions is the zero-one loss function defined as

$$c(\boldsymbol{x}, y_s, \mathcal{D}(\boldsymbol{x})) = l(\mathcal{D}(\boldsymbol{x})|y_s) = \begin{cases} 0 \text{ if } \mathcal{D}(\boldsymbol{x}) = y_s \\ 1 \text{ if } \mathcal{D}(\boldsymbol{x}) \neq y_s \end{cases} , \tag{2.7}$$

hence it penalizes only the incorrect classifications. The conditional risk becomes now

$$R(\mathcal{D}(\boldsymbol{x})|\boldsymbol{x}) = \sum_{q=1}^{Q} p(y_q|\boldsymbol{x}) - p(y_s|\boldsymbol{x}) = 1 - p(y_s|\boldsymbol{x}) \,. \tag{2.8}$$

It can be seen, that the conditional risk (so the overall risk) is minimized when the involved classifier $\mathcal{D}$ assigns a vector $\boldsymbol{x}$ to the class with maximal posterior probability given $\boldsymbol{x}$. Hence, the optimal classifier $\mathcal{D}^*$ is chosen according to the rule

$$\mathcal{D}^*(\boldsymbol{x}) = \underset{y_q \in \boldsymbol{Y}}{\arg\max}\{p(y_q|\boldsymbol{x})\}, \ \forall \boldsymbol{x} \in \boldsymbol{X} \,. \tag{2.9}$$

Such a rule is also known as *optimal Bayes* or *optimal Maximum A-Posteriori* (MAP) decision rule. Note that the overall risk represents now the probability of an error. The optimal Bayes decision rule is strictly optimal in the sense of minimizing the probability of an error, and only optimal for minimizing the overall risk subject to the 0-1 loss function [10]. An example of an optimal decision rule when three classes are present is shown in Figure 2.1.

Regrettably, prior and posterior distributions of $\boldsymbol{X}$ and $\boldsymbol{Y}$ are *unknown*. Therefore, none of the before mentioned equations can be evaluated. However, in real-life applications each problem goes along with some examples. If the examples would not exist, the problem would not exist as well (no information = no problem). Such examples are called training data and are the only submitted source of information. Thus, *only approximations* of the optimal classifier can be found, whereas the accuracy is strictly dependent on the quality of the training data (i.e. how well do they represent their parent class).

2.2 Parametric Classifiers

Parametric classifiers try to build a structure upon the data in the training set to learn the prior and posterior probabilities discussed in the previous section. The idea is to estimate a model (e.g. statistical) that represents each class, and compute the decision rule (2.9) indirectly utilizing the Bayes theorem, where the posterior probability for class y_q can be expressed as

$$p(y_q|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y_q)p(y_q)}{p(\boldsymbol{x})}, \tag{2.10}$$

and

$$p(\boldsymbol{x}) = \sum_{q=1}^{Q} p(\boldsymbol{x}|y_q)p(y_q) \tag{2.11}$$

is called the *evidence*. The decision rule (2.9) can be now rewritten into the form

$$\mathcal{D}^*(\boldsymbol{x}) = \underset{y_q \in \boldsymbol{Y}}{\arg\max} \ \{\ln p(\boldsymbol{x}|y_q) + \ln p(y_q)\} \tag{2.12}$$

and for equiprobable classes we get

$$\mathcal{D}^*(\boldsymbol{x}) = \underset{y_q \in \boldsymbol{Y}}{\arg\max} \ \{\ln p(\boldsymbol{x}|y_q)\} \ . \tag{2.13}$$

The entity $p(\boldsymbol{x}|y_q)$ is derived from the training data and is called the *class conditional density* or *class model*. The logarithmic function is involved because of the computational convenience. Classifiers based on class models are also denoted as *generative*. One of the benefits is that they can generate samples of the data through (2.11). We will focus on Gaussian Mixture Models (GMMs) and also the concept of Hidden Markov Models (HMMs) will be discussed.

2.2.1 Gaussian Mixture Model (GMM)

Gaussian Mixture Models were firstly introduced to the speaker recognition by Reynolds [11] and are widely used up to now. For a D dimensional feature vector $\boldsymbol{x}$ the GMM containing M Gaussians takes the form

$$g(\boldsymbol{x}) = p(\boldsymbol{x}|\boldsymbol{\lambda}) = \sum_{m=1}^{M} \omega_m p(\boldsymbol{x}|m, \boldsymbol{\lambda}) \, , \tag{2.14}$$

$$p(\boldsymbol{x}|m, \boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{C}_m) = \frac{1}{(2\pi)^{D/2} \left|\boldsymbol{C}_m\right|^{1/2}} \exp\left\{-0.5 \left(\boldsymbol{x} - \boldsymbol{\mu}_m\right)^{\mathrm{T}} \boldsymbol{C}_m^{-1} \left(\boldsymbol{x} - \boldsymbol{\mu}_m\right)\right\} \, , \tag{2.15}$$

where $\omega_m, \boldsymbol{\mu}_m, \boldsymbol{C}_m$ denote m^{th} Gaussian weight, mean and covariance, respectively, and

$$\boldsymbol{\lambda} = \{\omega_m, \boldsymbol{\mu}_m, \boldsymbol{C}_m\}_{m=1}^{M} \tag{2.16}$$

is the set of unknown parameters of the model. There are some restrictions laid on the Gaussian weights of a GMM in order to preserve the property of a probability distribution function. These can be expressed as

$$\forall m : 0 \leq \omega_m \leq 1 \ \text{ and } \ \sum_{m=1}^{M} \omega_m = 1 \, . \tag{2.17}$$

Generally, the covariance matrix $\boldsymbol{C}_m$ is considered full, nevertheless in praxis often diagonal matrices are assumed. This is caused especially because of numerical stability and computational costs, these

can occur mainly in situations when working in higher dimensions. After the class models have been trained, the label y_q in (2.13) is replaced by speaker specific model parameters $\boldsymbol{\lambda}_q$ and the classifier is brought to bear. GMMs are well suited for description of static (context-independent) data sources, where the time progress of samples is of no interest. In speaker recognition they are used mainly in text-independent tasks. They delimitate subspaces in the feature space that are characteristic for individual speakers. To train the GMM an iterative method called Expectation-Maximization (EM) algorithm may be exploited [12]. It is based on the Maximum Likelihood (ML) approach and tries to maximize the output probability of the model for submitted training data.

2.2.2 Hidden Markov Model (HMM)

Hidden Markov Models were developed in the 1960's by Baum and his colleagues [13] and have successfully spread to all the scientific branches. Now, the class to which a feature vector is assigned depends not only on the presented feature vector, but also on values of the other feature vectors and on relations among various classes [14]. There are several assumptions concerning the HMMs. Consider a sequence of classes $\Upsilon : y_1, \ldots, y_Q$, then the Markov model assumes that

- the class dependence is limited within two succesive classes; $p(y_q|y_{q-1}, \ldots, y_1) = p(y_q|y_{q-1})$,
- the feature vectors are statisticaly independent given Υ,
- the probability distributions in one class are independent of the other classes.

The principle of speech modelling according to the HMM comes from the idea that the arrangement of the vocal tract arises from a finite set of states, where each state corresponds to a distinct vocal tract configuration responsible for a specific sound signal. We will focus on HMMs with output probabilities of states represented by GMMs. Such models are also denoted as Continuous Density Hidden Markov Models (CDHMMs). Note that in this scenario the classes $y_1, \ldots, y_Q$ can represent distinct phonemes or some other phonetic events.

The Hidden Markov Model is characterized by a set of parameters $\boldsymbol{\Lambda} = \{\boldsymbol{U}, \boldsymbol{A}, \boldsymbol{G}, \boldsymbol{\pi}\}$, where $\boldsymbol{U} = \{u_1, \ldots, u_J\}$ represents the set of J states and $\boldsymbol{A} = [a_{ij}]$ stands for the matrix of state transitions. Elements in $\boldsymbol{A}$ determine the probability of being in the state i and subsequently moving to the state j, hence

$$a_{ij} = p(u(t+1) = u_j | u(t) = u_i), \;\; i, j \in \{1, \ldots, J\} \,. \tag{2.18}$$

The column vector $\boldsymbol{\pi} = [\pi_i]$ represents the initial state probabilities

$$\pi_i = P(u(1) = u_i) \,. \tag{2.19}$$

Sometimes even final state probabilities are defined. And at last, the set $\boldsymbol{G} = \{\boldsymbol{\lambda}_{u_1}, \ldots, \boldsymbol{\lambda}_{u_J}\}$ defines the GMM parameters (see (2.16)) for the states, though not each of the parameters has to be defined. States with defined probability distributions are called *emitting states*. The non-emitting states are used when several, separately trained HMMs have to be concatenated (e.g. for each monophone an individual HMM is trained and subsequently, the HMMs are concatenated utilizing the non-emitting states to model a word, sentence, etc.).

After the model parameters $\boldsymbol{\Lambda}$ have been estimated, the HMM output probability can be computed in the following way. Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be the set of N feature vectors, ψ denote a state-level

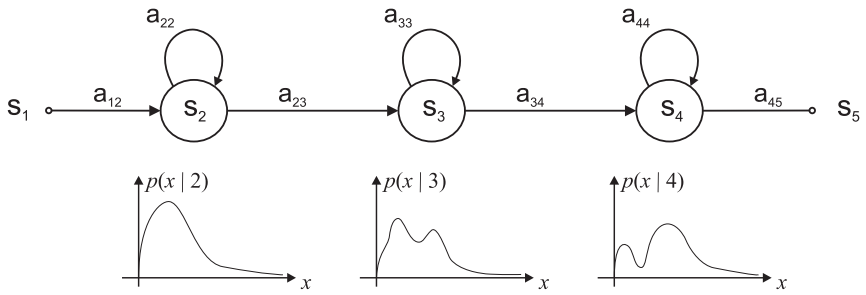

Figure 2.2: An example of a left-to-right, five state Hidden Markov Model with two non-emitting states s_1 and s_5. The transition probabilities are described by a_{ij} and the output probabilities of states s_2, s_3 and s_4 are represented by the GMMs (illustrated below each state).

path through the HMM and $\boldsymbol{\Psi}$ the entire set of such paths [10], then

$$\begin{aligned} p(\boldsymbol{X}|\boldsymbol{\Lambda}) &= \sum_{\psi \in \boldsymbol{\Psi}} p(\boldsymbol{X}, \psi|\boldsymbol{\Lambda}) = \sum_{\psi \in \boldsymbol{\Psi}} p(\boldsymbol{X}|\psi, \boldsymbol{\Lambda}) p(\psi|\boldsymbol{\Lambda}) = \\ &= \sum_{\psi \in \boldsymbol{\Psi}} \left(\prod_{k=1}^{N} g^{\psi}_{u(k)}(\boldsymbol{x}_k) \right) \left(\prod_{k=1}^{N} a^{\psi}_{u(k),u(k+1)} \right) \\ &= \sum_{\psi \in \boldsymbol{\Psi}} \left(\prod_{k=1}^{N} g^{\psi}_{s(k)}(\boldsymbol{x}_k)\, a^{\psi}_{u(k),u(k+1)} \right), \end{aligned} \tag{2.20}$$

where $g^{\psi}_{u(k)}(\boldsymbol{x}_k)$ is the state output probability defined in (2.14) and the upper index $^{\psi}$ indicates the presence in the state-level path ψ. Using directly (2.20) is computationally unbearable, therefore several efficient methods were developed, e.g. *forward-backward* or *Viterbi* algorithm (for details see [14]). Similarly to the GMM case, the classification is done according to

$$\boldsymbol{\Lambda}^* = \underset{\boldsymbol{\Lambda} \in L(\boldsymbol{\Lambda})}{\arg\max} \left\{ p(\boldsymbol{X}|\boldsymbol{\Lambda}) \right\}, \tag{2.21}$$

where $L(\boldsymbol{\Lambda})$ represents the set of parameters describing each participating HMM (classifier). To train the HMM an iterative Baum-Welch algorithm may be utilized. It is a ML parameter estimation procedure, basically similar to the EM algorithm. Regrettably a method that would lead to the global maximum was not found yet, hence it is wise to run the training algorithm several times with different initial conditions.

HMM classifiers are well suited and mostly exploited in the text-dependent recognition. However, they can be utilized also in the text-independent recognition in the form of so-called *ergodic models* [15]. In the task of speech recognition, the topology of an HMM depends mostly on the choice of the linguistic unit (e.g. triphone, monophone, sylable, word etc.) that statistical dependencies should be captured. An example of a left-to-right, five state HMM with two non-emitting states is depicted in Figure 2.2.

2.3 Nonparametric Classifiers

Such classifiers learn decision rules directly from the data. They do not involve class models and may be more accurate in cases when only a few training data are present. As the evidence (2.11) is not available, nonparametric classifiers cannot generate samples of the data. We will focus on linear discriminant functions of the form

$$\mathcal{H} : f(\boldsymbol{x}|\boldsymbol{\theta}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b, \;\; \boldsymbol{\theta} = \{\boldsymbol{w}, b\}, \tag{2.22}$$

where $\boldsymbol{w}$ is the normal vector of the hyperplane $\mathcal{H}$ and the scalar b denotes the offset, $\boldsymbol{\theta}$ are model parameters. Such a function divides the space into two half-spaces appropriate for two classes y_1 and y_2. The output value of the function (2.22) does not define the geometrical distance of the point $\boldsymbol{x}$ from the hyperplane $\mathcal{H}$, this can be acquired according to

$$d(\boldsymbol{x}, \mathcal{H}) = \frac{f(\boldsymbol{x})}{\|\boldsymbol{w}\|} \,. \tag{2.23}$$

The classification obeys the rule

$$\mathcal{D}(\boldsymbol{x}) = \begin{cases} y_1 \text{ if } f(\boldsymbol{x}) \geq 0 \\ y_2 \text{ if } f(\boldsymbol{x}) < 0 \end{cases} \,. \tag{2.24}$$

Of course, such linear discriminants behave well when the classes are linearly separable, but such an assumption is relatively rare and often nonlinear classifiers are demanded. It is quite difficult to ensure good generalization ability of nonlinear classifiers. However, utilizing training algorithms that involve so-called *kernels*, one can extend linear classifiers also to nonlinear cases. The kernel trick may be exploited in all the algorithms that approach the data only through dot products. Another question concerns the requirement laid on the choice of the separating hyperplane. All the before stated demands are solved in the concept of the Support Vector Machine (SVM).

2.3.1 Support Vector Machine (SVM)

Support Vector Machine was firstly introduced by Vapnik [16]. The requirement set on the decision hyperplane $\mathcal{H}$ concerns the width of the margin between the two classes that should be separated. Let us adjust (2.24) in the manner of (2.25).

$$\begin{array}{l} \mathcal{H}_1 : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \geq +1 \text{ for } y_i = +1 \\ \mathcal{H}_2 : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \leq -1 \text{ for } y_i = -1 \end{array} \quad \Longrightarrow \quad \forall i : y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1 \geq 0 \,. \tag{2.25}$$

Hence, we have requested a margin between the two classes y_1 and y_2. The width of the margin can be easily computed using (2.23). Thus, $|d(\mathbf{0}, \mathcal{H}_1) - d(\mathbf{0}, \mathcal{H}_2)| = 2/\|\boldsymbol{w}\|$, where $\mathbf{0}$ represents a zero vector (origin). The formulation of SVM demands the widest margin, therefore we are seeking for $\boldsymbol{w}$ with the minimal norm. To allow errors (points that violate the decision boundaries $\mathcal{H}_1$ and $\mathcal{H}_2$) and to relax the constraint of strict class pertinence in (2.25), Vapnik introduced non-negative variables ξ_i denoted the *slack variables*. The constraints become now

$$\begin{array}{l} \mathcal{H}_1 : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \geq +1 - \xi_i \text{ for } y_i = +1 \,, \\ \mathcal{H}_2 : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \leq -1 + \xi_i \text{ for } y_i = -1 \,, \\ \xi_i \geq 0 \,. \end{array} \tag{2.26}$$

Thus, when an error occures, the slack variable exceeds unity and $\sum_i \xi_i$ will measure the upper bound of errors, otherwise ξ_i remains zero. The term $\sum_i \xi_i$ represents the cost (loss) defined in (2.3). Now, the problem can be formulated as

$$\text{minimize} \quad \left[\frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_i \xi_i \right] \,, \tag{2.27}$$

subject to (2.26), where C is an additional term defined by the user to set a cost for errors. There are a lot of other possibilities how to choose the loss function, e.g. squared-error cost $\sum_i \xi_i^2$ (for other choices see [17]). An example of the SVM composition is depicted in Figure 2.3. Note that the square of $\boldsymbol{w}$

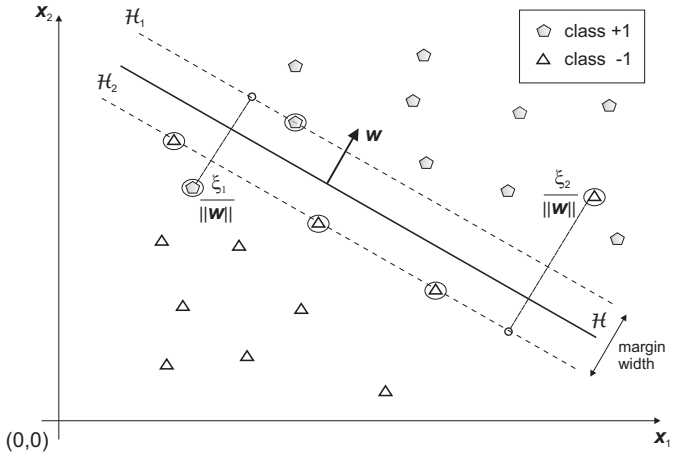

Figure 2.3: An example of the non-separable case of the SVM problem. Vectors encapsulated in circles denote support vectors.

was involved in order to ensure convexity of the proposed problem and to guarantee only one, *globally optimal* solution. To solve the problem of constrained convex optimization, Lagrange multipliers are involved, whereas the solution is obtained in its dual form (for details see [18]). Another handy tool providing useful informations about the solution are the Karush-Kuhn-Tucker (KKT) conditions, well-known in the nonlinear programming. The Lagrange formula of the primal problem with Lagrange multipliers α_i and β_i can be expressed as

$$L_{\mathrm{P}} = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\{y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1 + \xi_i\} - \sum_i \beta_i\xi_i\,. \tag{2.28}$$

Solving the problem (2.28) (seeking for mimimum) results in

$$\begin{aligned}
\frac{\partial L_{\mathrm{P}}}{\partial \boldsymbol{w}} &= \boldsymbol{w} - \sum_i \alpha_i y_i \boldsymbol{x}_i &&= 0\,,\\
\frac{\partial L_{\mathrm{P}}}{\partial b} &= -\sum_i \alpha_i y_i &&= 0\,,\\
\frac{\partial L_{\mathrm{P}}}{\partial \xi_i} &= C - \alpha_i - \beta_i &&= 0\,,
\end{aligned} \tag{2.29}$$

with additional KKT conditions

$$\begin{aligned}
y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1 + \xi_i &&&\geq 0\,,\\
\alpha_i\{y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1 + \xi_i\} &&&= 0\,,\\
\beta_i\xi_i &&&= 0\,,\\
\xi_i, \alpha_i, \beta_i &&&\geq 0\,.
\end{aligned} \tag{2.30}$$

Substituting (2.29) to (2.28) gives us the dual formulation, which has to be *maximized*, in the form of

$$L_{\mathrm{D}} = \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j \tag{2.31}$$

subject to (2.30). According to (2.29) the normal vector $\boldsymbol{w}$ can be computed as

$$\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i\,. \tag{2.32}$$

Thus, the decision hyperplane (2.22) results in

$$\mathcal{H} : f(\boldsymbol{x}) = \left(\sum_i \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}} \right) \boldsymbol{x} + b = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b \,. \tag{2.33}$$

After inspection of the KKT conditions it is quite clear that $\alpha_i \in (0, C)$ (consider the third condition in (2.29) and fourth condition in (2.30)). Furthermore, the second condition in (2.30) implies that α_i is zero for all the vectors that lie on the correct side of the margin, and nonzero only for vectors that violate the margin ($\xi_i > 0$) or vectors lying on the margin generated by hyperplanes $\mathcal{H}_1$ and $\mathcal{H}_2$ (defined in (2.26)). Vectors with nonzero α_i are called the *support vectors* as just they participate in (2.32). The offset b cannot be computed directly – it does not occur in (2.29). However, it can be computed utilizing the second KKT condition in (2.30) choosing any i for which $\alpha_i \neq 0$, but it is numerically safer to take the mean value of b resulting from all such equations [18].

Note that equations (2.31) and (2.33) depend on feature vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ only through their dot products. Consider a mapping $\Phi : \boldsymbol{X} \mapsto \boldsymbol{\mathcal{X}}$, where $\boldsymbol{\mathcal{X}}$ represents some Euclidean feature space (even infinite dimensional). Then, all the dot products in (2.31) and (2.33) may be replaced by dot products $\Phi(\boldsymbol{x}_i)\Phi(\boldsymbol{x}_j)$ defined in the space $\boldsymbol{\mathcal{X}}$. Furthermore, utilizing a function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)\Phi(\boldsymbol{x}_j)$ all that matters is the output of such a *kernel function* $K(\cdot, \cdot)$, the individual mappings $\Phi(\boldsymbol{x}_i)$ become *unnecessary*. Now, the two equations (2.31), (2.33) can be rewritten as

$$L_{\mathrm{D}} = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \,, \tag{2.34}$$

$$\mathcal{H} : f(\boldsymbol{x}) = \sum_i \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b \,. \tag{2.35}$$

It should be stated that the matrix

$$\boldsymbol{H} = [H_{ij}], \;\; H_{ij} = \frac{\partial^2 L_{\mathrm{D}}}{\partial \alpha_i \partial \alpha_j} = -y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{2.36}$$

represents the Hessian matrix, which is used to determine the global maximum of the optimization problem (2.34).

Some of the most popular kernel functions are

- simple linear kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$,
- general polynomial kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (a \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + c)^p$,
- Radial Basis Function (RBF) kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \left| \boldsymbol{x}_i - \boldsymbol{x}_j \right|^2)$, $\gamma > 0$.

Of course, many other kernels were developed and tested through the time (kernels utilized in speaker recognition will be in more depth discussed in Chapter 4). Generally, any function representing a dot product in some space may be considered as a kernel function. The condition that a function has to satisfy to be a valid kernel is known as *Mercer's condition* [16]. Suppose any square integrable function $f(\boldsymbol{x})$,

$$\int f(\boldsymbol{x})^2 \mathrm{d}\boldsymbol{x} < \infty \,. \tag{2.37}$$

Then $K(\boldsymbol{x}, \boldsymbol{z})$ is a valid kernel if and only if

$$\int K(\boldsymbol{x}, \boldsymbol{z}) f(\boldsymbol{x}) f(\boldsymbol{z}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{z} \geq 0 \,. \tag{2.38}$$

The main aspects of the Mercer's condition are that the kernel function has to be symmetric ($K(\boldsymbol{x},\boldsymbol{z}) = K(\boldsymbol{z},\boldsymbol{x})$) and non-negative definite.

To train the SVM nonlinear programming techniques are utilized. Many trainers have been already developed and coded, e.g. Thorsten's SVM$^{\text{light}}$ [19] suitable for sparse data problems (e.g. derivative kernels – see Section 4.4) or also very popular SVMtorch [20] and LibSVM [21].

Multi-class Problem SVM is inherently a binary classifier. The pertinence to a class is established in accordance to the position of a vector on one of the sides of the separating hyperplane, see (2.24). When several classes are given *one-against-all* training can be utilized, where in the training process all the examples from other classes are pulled together and used as negative examples. For each of the classes $y_q, q = 1,\ldots,Q$ one hyperplane $\boldsymbol{\theta}_q = \{\boldsymbol{w}_q, b_q\}$ is estimated, and the classification of an unknown vector $\boldsymbol{x}$ is given as

$$y_q = \mathcal{D}(\boldsymbol{x}) = \arg\max_q \left(\boldsymbol{w}_q^{\mathrm{T}}\boldsymbol{x} + b_q\right). \tag{2.39}$$

Another way to train the SVM is the *all-pairs* technique described in [22]. For each pair of classes a binary classifier is trained and the pairwise results are combined at the end. Hence, during the training of a decision boundary between two different classes the data from other classes are ignored. Now, for each class y_q we get Q hyperplanes (decision boundaries). Obviously, the main disadvantage of such an approach is the need to evaluate the relative distance (2.22) of a vector from all of the Q hyperplanes when classifying an unknown vector. This may be computationally expensive in cases when many classes are present.

Dealing with Unbalanced Data The problem arises when the amount of positive and negative examples submitted to the SVM training is highly disproportional. This occurs mainly when models for several classes are trained and one-against-all training is utilized. One of the solutions is to implement a weighting of misclassified examples directly in the formulation of the SVM problem [23], thus the formulation of the problem changes to

$$\text{minimize} \quad \left[\frac{1}{2}\|\boldsymbol{w}\|^2 + C^+\sum_{i:y_i=+1}\xi_i + C^-\sum_{j:y_j=-1}\xi_j\right] \tag{2.40}$$

subject to

$$\forall i : y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1 - \xi_i\,. \tag{2.41}$$

Hence, we are able to adjust the cost of misclassifying positive and negative examples separately – the setting $C^+ > C^-$ implies that greater cost is set on misclassifying positive examples and vice versa. Now, the class with less examples should get higher penalization.

Probabilistic Outputs for SVM The probability outputs are very handy in situations when several systems are combined in order to obtain an overall decision. The result of the SVM classifier is the relative distance from the decision boundary and it is in principle unbounded. Thus, it can take any values in the interval $(-\infty,\infty)$, what makes the fusion with other systems ambiguous.

In order to get a probabilistic output of a SVM one can use a GMM to rescale the results [24]. The posterior probabilities $p(f(\boldsymbol{x}_i)|y_q = +1)$ and $p(f(\boldsymbol{x}_i)|y_q = -1)$ are estimated utilizing the ML approach and labeled data. The final decision is computed according to the Bayes rule

$$p(y_q = +1|f(\boldsymbol{x}_i)) = \frac{p(f(\boldsymbol{x}_i)|y_q = +1)P(y = +1)}{\sum_{j\in\{1,-1\}} p(f(\boldsymbol{x}_i)|y_q = j)P(y = j)}\,, \tag{2.42}$$

where $f(\boldsymbol{x}_i)$ represents the SVM hyperplane, y_q represents the class label of $\boldsymbol{x}_i$, and priors $P(y = \pm 1)$ are estimated on a training set.

In [25] a frequently used sigmoidal function is used. The probability $p(y_q = +1|f(\boldsymbol{x}_i))$ is estimated as

$$p(y_q = +1|f(\boldsymbol{x}_i)) = \frac{1}{1 + \exp(af(\boldsymbol{x}) + b)}\,, \tag{2.43}$$

where a, b control the slope and the position of the inflection point of the sigmoid, respectively. Both can be estimated using ML, see [25].

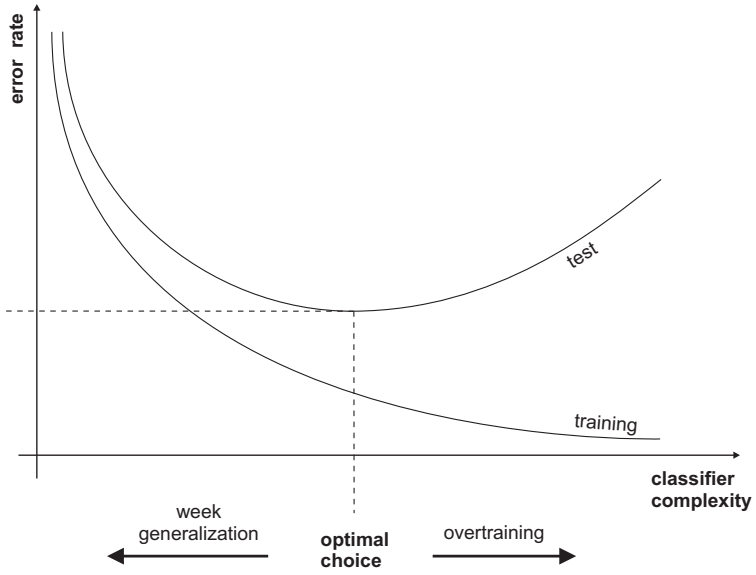

Figure 2.4: Overtraining phenomena.

2.4 Conclusion and Remarks

When a classifier is proposed and trained, it is very important to have appropriate testing data, which were not seen during the training. Consider a GMM with gradually increasing number of Gaussians. The situation is illustrated in Figure 2.4. At the beginning, the error rates acquired on test and training data will decrease, e.g. Gaussian mixtures with diagonal covariances will try to approximate the full-covariance case. But when too many Gaussians are employed, the Gaussians will unnecessarily oppress each other, each Gaussian will cover less training data (its variance will decrease), and generalization to unseen data will diminish. Such a problem is known as *overtraining*. The proper choice of the number of GMM mixtures is task dependent, relevant factors are dimensionality and amount of training data.

In this chapter only three classifiers, important for the successive explanation of the thesis objectives, were presented. Some other successfully exploited parametric classifiers are Markov Random Fields (MRFs) or Bayesian Networks (BNs), moreover the Chapter 6 will be devoted to a special class of generative models based on factor analysis. Systems based on these models represent a state-of-the-art in the speaker recognition. In the case of non-parametric classifiers popular Neural Networks (NN) and Logistic Regression (LR) should be mentioned (for details see e.g. [14]).

Chapter 3

Adaptation Techniques

The main assumption when using an adaptation technique is that some reliable/robust model was trained, and we wish to update its parameters according to given data leading to a reasonable model even in cases when the amount of training/adaptation data is small (one could not train a new reliable model from the scratch).

Techniques presented in this chapter will adjust the model so that the probability of the adaptation data would be maximized. Assuming a set of model parameters $\boldsymbol{\lambda}$ we can write

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} p(\boldsymbol{\lambda}|\boldsymbol{O}^1, \ldots, \boldsymbol{O}^S) = \arg\max_{\boldsymbol{\lambda}} p(\boldsymbol{O}^1, \ldots, \boldsymbol{O}^S|\boldsymbol{\lambda}) p(\boldsymbol{\lambda}), \tag{3.1}$$

where $\boldsymbol{O}^s = \{\boldsymbol{o}_1^s, \boldsymbol{o}_2^s, \ldots, \boldsymbol{o}_T^s\}, s = 1, \ldots, S$, is the sequence of feature vectors related to some measurement s, and $\boldsymbol{\lambda}^*$ is the best estimate of the model parameters given the feature vectors. Note that $p(\boldsymbol{\lambda})$ stands for the prior of model parameters $\boldsymbol{\lambda}$. In case where the prior is unknown (e.g. parameters are distributed uniformly) we speak about *Maximum Likelihood* (ML) estimation, otherwise we speak about *Maximum A-posteriori Probability* (MAP) training. The prior is useful mainly in situations, where the amount of input data is low – ML training would give inaccurate estimates. However, in following sections also ML training procedures will be described leading to good estimates even in situations when only a small amount of data for training is available. These techniques are based on restricting the number of free model parameters to be estimated via clustering of model parameters.

From now on we will focus on Gaussian Mixture Models (GMMs) described in Section 2.2.1, where the set $\boldsymbol{\lambda} = \{\omega_m, \boldsymbol{\mu}_m, \boldsymbol{C}_m\}_{m=1}^M$ is the set of GMM parameters, $\omega_m, \boldsymbol{\mu}_m, \boldsymbol{C}_m$ is the weight, mean, and covariance matrix of the m^{th} Gaussian, respectively. Note that instead of a full covariance matrix $\boldsymbol{C}_m$ often only its diagonal $\boldsymbol{\sigma}_m^2 = \text{diag}(\boldsymbol{C}_m)$ is estimated, where the function $\text{diag}(\boldsymbol{X})$ transforms a matrix to a vector taking only the diagonal of the input matrix $\boldsymbol{X}$. The set $\boldsymbol{O}^s$ will from now on represent the set of feature vectors related to the s^{th} speaker. The model $\boldsymbol{\lambda}^*$ defined in (3.1) is often denoted as Universal Background Model (UBM) [26], it is trained using the ML approach from a huge amount of data acquired from a large set of speakers, therefore the estimates of UBM parameters are sufficiently accurate. The speakers data should match all the conditions, in which the recognition system is going to be used. Then, the UBM is adapted to a small population of feature vectors of a speaker in order to get a Speaker Dependent (SD) model.

Since GMMs incorporate latent variables (the assignment of a feature vector to a Gaussian is unknown) Expectation Maximization (EM) algorithm is utilized, where the following auxiliary function is optimized

$$Q_{\text{ML}}(\boldsymbol{\lambda}, \bar{\boldsymbol{\lambda}}) = \sum_{m=1}^{M} \sum_{t=1}^{T} p(m|\boldsymbol{o}_t, \boldsymbol{\lambda}) \log p(\boldsymbol{o}_t, m|\bar{\boldsymbol{\lambda}}) \,. \tag{3.2}$$

This is the case in the ML estimation. MAP adds an additional term that represents the contribution

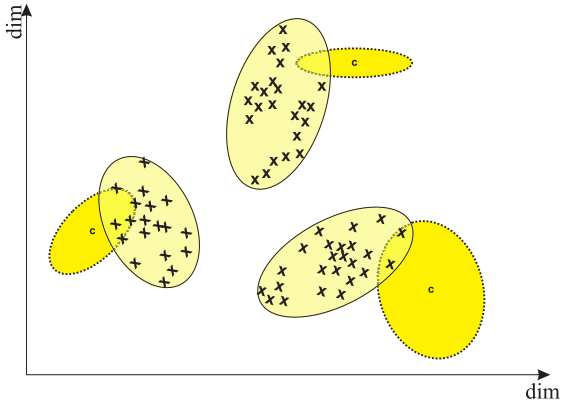

Figure 3.1: Adaptation of 3 Gaussians according to given data. The Gaussians of the initial GMM to be adapted (UBM) are shown as yellow ellipsoids with dotted borders. Adaptation data are represented by the x-marks, and ellipsoids with solid borders represent the adapted Gaussians. Ellipsoids express the distance from the mean of the Gaussian given as a multiple of the standard deviation.

from the prior distribution, hence

$$Q_{\mathrm{MAP}}(\boldsymbol{\lambda}, \bar{\boldsymbol{\lambda}}) = Q_{ML}(\boldsymbol{\lambda}, \bar{\boldsymbol{\lambda}}) + \log p(\bar{\boldsymbol{\lambda}})\,. \tag{3.3}$$

The adaptation principle of GMM composed of 3 Gaussian is depicted in Figure 3.1.

3.1 Adaptation Statistics

Equations (3.4) – (3.6) are common for all the adaptation techniques and we will refer to them in the consequent text. They represent the zero, first and second moments of given feature vectors aligned to individual Gaussians. It should be stated that all of the adaptation techniques approach the data only through the statistics introduced below. Hence, the adaptation process can be divided into two stages. In the first stage, the statistics are accumulated till the data are introduced to the algorithm, and in the second stage the desired type of adaptation is applied. These statistics are

$$\gamma_m(t) = p(m|\boldsymbol{o}_t, \boldsymbol{\lambda}) = \frac{\omega_m p(\boldsymbol{o}_t|m, \boldsymbol{\lambda})}{\sum_{m=1}^{M} \omega_m p(\boldsymbol{o}_t|m, \boldsymbol{\lambda})}, \tag{3.4}$$

$$c_m = \sum_{t=1}^{T} \gamma_m(t), \tag{3.5}$$

$$\boldsymbol{\varepsilon}_m(\boldsymbol{o}) = \frac{1}{c_m} \sum_{t=1}^{T} \gamma_m(t)\boldsymbol{o}_t, \;\; \boldsymbol{\varepsilon}_m^2(\boldsymbol{o}) = \frac{1}{c_m} \sum_{t=1}^{T} \gamma_m(t)\boldsymbol{o}_t\boldsymbol{o}_t^{\mathrm{T}}, \tag{3.6}$$

the posterior probability of m^{th} Gaussian given a feature vector $\boldsymbol{o}_t$, the zero (soft count), the first and the second moment of feature vectors aligned to the m^{th} Gaussian, respectively.

3.2 Maximum A-Posteriori Probability (MAP) Adaptation

MAP adapts each of the GMM parameters from the set $\boldsymbol{\lambda} = \{\omega_m, \boldsymbol{\mu}_m, \boldsymbol{C}_m\}_{m=1}^{M}$ separately. Thus, it is necessary to have enough adaptation data for all the parameters, otherwise would be the result of adaptation negligible (but the new model estimate would be still robust). The new parameters

$\bar{\boldsymbol{\lambda}} = \{\bar{\omega}_m, \bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{C}}_m\}_{m=1}^{M}$ are acquired by optimizing the objective function (3.3) [27]. This leads to update formulas

$$\bar{\omega}_m = [\alpha_m c_m/T + (1-\alpha_m)\omega_m]\,\chi\,, \tag{3.7}$$

$$\bar{\boldsymbol{\mu}}_m = \alpha_m \boldsymbol{\varepsilon}_m(\boldsymbol{o}) + (1-\alpha_m)\boldsymbol{\mu}_m\,, \tag{3.8}$$

$$\bar{\boldsymbol{C}}_m = \alpha_m \boldsymbol{\varepsilon}_m^2(\boldsymbol{o}) + (1-\alpha_m)(\boldsymbol{C}_m + \boldsymbol{\mu}_m\boldsymbol{\mu}_m^{\mathrm{T}}) - \bar{\boldsymbol{\mu}}_m\bar{\boldsymbol{\mu}}_m^{\mathrm{T}}\,, \tag{3.9}$$

where

$$\alpha_m = \frac{c_m}{c_m + \tau}\,. \tag{3.10}$$

Hence, α_m represents the adaptation coefficient (*relevance factor*) that controls the balance between old and new parameters using the parameter τ. The parameter τ is set by the user and determines the amount of new data that have to match a specific Gaussian till the Gaussian parameters change (they shift in the direction of new parameters) [28]. χ is a normalization factor, which guarantees that all the new weights of a GMM sum to one.

3.3 Maximum Likelihood Linear Regression (MLLR)

In contrast to the MAP adaptation, where large amount of data is needed in order to change the values of GMM parameters, MLLR reduces the number of available model parameters via clustering (commonly used are regression trees) of similar components. Parameters from the same cluster/class $K_n, n = 1, \ldots, N$, share the same transformation matrix. Thus, less data are needed for MLLR to be effective. Now, the ML objective function (3.2) is maximized, hence none a-priori information is utilized. It can be rewritten to the form

$$Q_{\mathrm{ML}}(\boldsymbol{\lambda}, \bar{\boldsymbol{\lambda}}) = \mathrm{const} - \frac{1}{2}\sum_m\sum_t \gamma_m(t)\left[\mathrm{const}_m + \log|\bar{\boldsymbol{C}}_m| + (\boldsymbol{o}_t - \bar{\boldsymbol{\mu}}_m)^{\mathrm{T}}\bar{\boldsymbol{C}}_m^{-1}(\boldsymbol{o}_t - \bar{\boldsymbol{\mu}}_m)\right]\,, \tag{3.11}$$

where $\bar{\boldsymbol{\lambda}} = \{\omega_m, \bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{C}}_m\}_{m=1}^{M}$, thus Gaussian weights are not adapted, and const, const_m denote terms independent of $\bar{\boldsymbol{\lambda}}$. Also note that the presence of the model $\boldsymbol{\lambda}$ persists in the posterior $\gamma_m(t)$ (in order to compute $\gamma_m(t)$ model parameters $\boldsymbol{\lambda}$ are used). The task is to find linear transformations of GMM means and GMM variances that would maximize the auxiliary function specified in (3.11), where we assume that means are transformed according to the formula

$$\bar{\boldsymbol{\mu}}_m = \boldsymbol{A}_{(n)}\boldsymbol{\mu}_m + \boldsymbol{b}_{(n)} = \boldsymbol{W}_{(n)}\boldsymbol{\xi}_m,\ m \in K_n, \tag{3.12}$$

$$\boldsymbol{W}_{(n)} = [\boldsymbol{A}_{(n)}, \boldsymbol{b}_{(n)}],\ \ \boldsymbol{\xi}_m = [\boldsymbol{\mu}_m^{\mathrm{T}}, 1]^{\mathrm{T}}, \tag{3.13}$$

where $\boldsymbol{\mu}_m$ is the UBM mean of the m^{th} Gaussian classified to the cluster K_n, $\bar{\boldsymbol{\mu}}_m$ is the new adapted mean, $\boldsymbol{A}_{(n)}$ and $\boldsymbol{b}_{(n)}$ are the adaptation matrix and the additive vector related to the n^{th} cluster K_n, respectively. And the adaptation formula for the new covariance matrix $\bar{\boldsymbol{C}}_m$ is

$$\bar{\boldsymbol{C}}_m = \boldsymbol{H}_{(n)}\boldsymbol{C}_m\boldsymbol{H}_{(n)}^{\mathrm{T}}, \tag{3.14}$$

where again $\boldsymbol{H}_{(n)}$ is associated with the cluster K_n. Note that the covariance matrix $\boldsymbol{C}_m$ and the mean $\boldsymbol{\mu}_m$ of the m^{th} Gaussian do not have to be necessarily in the same cluster K_n (this depends on how the regression tree is constructed and what kind of similarity measure is used in the clustering process).

The part of the objective function (3.11) that changes with the current transform $\boldsymbol{W}_{(n)}$ can be written as [29]

$$Q_{\boldsymbol{W}_{(n)}} = \mathrm{const} - \sum_{m\in K_n} c_m \sum_{i=1}^{D} \frac{(\boldsymbol{w}_{(n)i}^{\mathrm{T}}\boldsymbol{\xi}_m)^2 - 2(\boldsymbol{w}_{(n)i}^{\mathrm{T}}\boldsymbol{\xi}_m)\varepsilon_{mi}(\boldsymbol{o})}{\sigma_{mi}^2}\,, \tag{3.15}$$

where the column vector $\boldsymbol{w}_{(n)i}$ equals the transpose of the i^{th} row of $\boldsymbol{W}_{(n)}$ and D is the dimension of feature vectors. Equation (3.15) can be further rearranged to the form

$$Q_{\boldsymbol{W}_{(n)}} = \sum_{i=1}^{D} \left(\boldsymbol{w}_{(n)i}^{\mathrm{T}} \boldsymbol{k}_{(n)i} - 0.5\, \boldsymbol{w}_{(n)i}^{\mathrm{T}} \boldsymbol{G}_{(n)i} \boldsymbol{w}_{(n)i} \right) , \tag{3.16}$$

where the const part was omitted and (3.15) was divided by the factor of 2, and

$$\boldsymbol{k}_{(n)i} = \sum_{m \in K_n} \frac{c_m \boldsymbol{\xi}_m \varepsilon_{mi}(\boldsymbol{o})}{\sigma_{mi}^2} \tag{3.17}$$

and

$$\boldsymbol{G}_{(n)i} = \sum_{m \in K_n} \frac{c_m \boldsymbol{\xi}_m \boldsymbol{\xi}_m^{\mathrm{T}}}{\sigma_{mi}^2} . \tag{3.18}$$

And finally the maximization of equation (3.16) gives us the updating formulas

$$\frac{\partial Q(\boldsymbol{\lambda}, \bar{\boldsymbol{\lambda}})}{\partial \boldsymbol{W}_{(n)}} = 0 \Rightarrow \boldsymbol{w}_{(n)i} = \boldsymbol{G}_{(n)i}^{-1} \boldsymbol{k}_{(n)i} . \tag{3.19}$$

The transformation equations for covariance matrices can be derived in analogy with the previous approach. They will be not discussed here as they are no important for further explanations and can be found in [30].

3.4 Feature MLLR (fMLLR) and Constrained MLLR (CMLLR)

Compared to MLLR the transformation is now applied on the feature space (feature MLLR) instead of on model parameters. The objective function (3.2) changes to [30]

$$Q(\boldsymbol{\lambda}, \bar{\boldsymbol{\lambda}}) = \text{const} - \frac{1}{2} \sum_m \sum_t \gamma_m(t) [\text{const}_m + \log |\boldsymbol{C}_m| - \log |\boldsymbol{A}_{(n)}|^2 + \tag{3.20}$$

$$+ (\bar{\boldsymbol{o}}_t - \boldsymbol{\mu}_m)^{\mathrm{T}} \boldsymbol{C}_m^{-1} (\bar{\boldsymbol{o}}_t - \boldsymbol{\mu}_m)] . \tag{3.21}$$

The feature vectors are transformed according to the formula

$$\bar{\boldsymbol{o}}_t = \boldsymbol{A}_{(n)} \boldsymbol{o}_t + \boldsymbol{b}_{(n)} = \boldsymbol{A}_{(n)c}^{-1} \boldsymbol{o}_t + \boldsymbol{A}_{(n)c}^{-1} \boldsymbol{b}_{(n)c} = \boldsymbol{W}_{(n)} \boldsymbol{\xi}(t) , \tag{3.22}$$

where $\boldsymbol{W}_{(n)} = [\boldsymbol{A}_{(n)}, \boldsymbol{b}_{(n)}]$ is the transformation matrix corresponding to the n^{th} cluster/class K_n, $\boldsymbol{\xi}(t) = [\boldsymbol{o}_t^{\mathrm{T}}, 1]^{\mathrm{T}}$ represents the extended feature vector. It can be shown [30] that the transformation performed on features may be replaced by an *equivalent transformation* performed on model means and covariances utilizing matrices $\boldsymbol{A}_{(n)\mathrm{c}} = \boldsymbol{A}_{(n)}^{-1}$ and $\boldsymbol{b}_{(n)\mathrm{c}} = \boldsymbol{A}_{(n)}^{-1} \boldsymbol{b}_{(n)}$. Hence, model parameters can be transformed using formulas

$$\bar{\boldsymbol{\mu}}_m = \boldsymbol{A}_{(n)\mathrm{c}} \boldsymbol{\mu}_m - \boldsymbol{b}_{(n)\mathrm{c}} , \tag{3.23}$$

and

$$\bar{\boldsymbol{C}}_m = \boldsymbol{A}_{(n)\mathrm{c}} \boldsymbol{C}_m \boldsymbol{A}_{(n)\mathrm{c}}^{\mathrm{T}} . \tag{3.24}$$

This method is known as Constrained MLLR (CMLLR), because the same transformation matrix is used for means as well as for covariances. Loosely speaking, fMLLR and CMLLR are *equivalent*

transformations and the only difference consists in their interpretation. In analogy with the previous section, it is possible to rearrange the objective function (3.21) to the form [29]

$$Q_{\boldsymbol{W}_{(n)}}(\boldsymbol{\lambda}, \bar{\boldsymbol{\lambda}}) = \beta_{(n)} \log |\boldsymbol{A}_{(n)}| + \sum_{i=1}^{D} \left(\boldsymbol{w}_{(n)i}^{\mathrm{T}} \boldsymbol{k}_i - \frac{1}{2} \boldsymbol{w}_{(n)i}^{\mathrm{T}} \boldsymbol{G}_{(n)i} \boldsymbol{w}_{(n)i} \right) , \tag{3.25}$$

where D is the dimension of feature vectors $\boldsymbol{o}_t$, and

$$\beta_{(n)} = \sum_{m \in K_n} \sum_{t} \gamma_m(t) , \tag{3.26}$$

$$\boldsymbol{k}_{(n)i} = \sum_{m \in K_n} \frac{c_m \mu_{mi} \varepsilon_m(\boldsymbol{\xi})}{\sigma_{mi}^2} , \tag{3.27}$$

$$\boldsymbol{G}_{(n)i} = \sum_{m \in K_n} \frac{c_m \varepsilon_m^2(\boldsymbol{\xi})}{\sigma_{mi}^2} , \tag{3.28}$$

$$\varepsilon_m(\boldsymbol{\xi}) = \left[\varepsilon_m^{\mathrm{T}}(\boldsymbol{o}), 1 \right]^{\mathrm{T}} , \tag{3.29}$$

$$\varepsilon_m^2(\boldsymbol{\xi}) = \begin{bmatrix} \varepsilon_m(\boldsymbol{o}\boldsymbol{o}^{\mathrm{T}}) & \varepsilon_m(\boldsymbol{o}) \\ \varepsilon_m(\boldsymbol{o})^{\mathrm{T}} & 1 \end{bmatrix} . \tag{3.30}$$

In order to find the solution of equation (3.25) we have to express $\boldsymbol{A}_{(n)}$ in terms of $\boldsymbol{W}_{(n)}$ (realize that $\boldsymbol{W}_{(n)} = [\boldsymbol{A}_{(n)}, \boldsymbol{b}_{(n)}]$). One of the possible solutions is the use of the equivalency $\log |\boldsymbol{A}_{(n)}| = \log |\boldsymbol{w}_{(n)i}^{\mathrm{T}} \boldsymbol{v}_{(n)i}|$, where $\boldsymbol{v}_{(n)i}$ stands for transpose of the i^{th} row of cofactors of the matrix $\boldsymbol{A}_{(n)}$ extended with a zero in the last dimension. Let $\alpha_{(n)i} = \boldsymbol{w}_{(n)i}^{\mathrm{T}} \boldsymbol{v}_{(n)i}$. After the maximization of the auxiliary function (3.25) we receive

$$\boldsymbol{w}_{(n)i} = \boldsymbol{G}_{(n)i}^{-1} \left(\frac{\boldsymbol{v}_{(n)i}}{\alpha_{(n)i}} + \boldsymbol{k}_{(n)i} \right) , \tag{3.31}$$

where $\alpha_{(n)}$ can be found as the solution of the quadratic function

$$\beta_{(n)} \alpha_{(n)i}^2 - \alpha_{(n)i} \, \boldsymbol{v}_{(n)i}^{\mathrm{T}} \boldsymbol{G}_{(n)i}^{-1} \boldsymbol{k}_{(n)i} - \boldsymbol{v}_{(n)i}^{\mathrm{T}} \boldsymbol{G}_{(n)i}^{-1} \boldsymbol{v}_{(n)i} = 0 . \tag{3.32}$$

Because the equation (3.32) is quadratic, two different solutions $\boldsymbol{w}_{(n)i}^1, \boldsymbol{w}_{(n)i}^2$ are obtained in (3.31). The one that maximizes the auxiliary function (3.25) is chosen.

Note that due to the fact that in fMLLR the transformation function $f(\boldsymbol{o}_t) = \boldsymbol{A}_{(n)} \boldsymbol{o}_t + \boldsymbol{b}_{(n)}$ is applied directly on feature vectors, an additional term – Jacobian of $f(\boldsymbol{o}_t)$ – will occur in the log likelihood related to each Gaussian [30]. Hence, for CMLLR we get the log likelihood

$$\log p\left(\boldsymbol{o}_t | \boldsymbol{\mu}_m, \boldsymbol{C}_m, \boldsymbol{A}_{(n)c}, \boldsymbol{b}_{(n)c}\right) = \log \mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{A}_{(n)c} \boldsymbol{\mu}_m - \boldsymbol{b}_{(n)c}, \boldsymbol{A}_{(n)c} \boldsymbol{C}_m \boldsymbol{A}_{(n)c}^{\mathrm{T}}\right) , \tag{3.33}$$

but for fMLLR we get

$$\log p\left(\boldsymbol{o}_t | \boldsymbol{\mu}_m, \boldsymbol{C}_m, \boldsymbol{A}_{(n)}, \boldsymbol{b}_{(n)}\right) = \log \mathcal{N}\left(\boldsymbol{A}_{(n)} \boldsymbol{o}_t + \boldsymbol{b}_{(n)}; \boldsymbol{\mu}_m, \boldsymbol{C}_m\right) + 0.5 \log |\boldsymbol{A}_{(n)}|^2 . \tag{3.34}$$

Another difference from MLLR is that the estimation of $\boldsymbol{W}_{(n)}$ in fMLLR becomes iterative since when computing $\boldsymbol{w}_{(n)i}$ in (3.31) both $\boldsymbol{v}_{(n)i}$ and $\alpha_{(n)i}$ does depend on $\boldsymbol{w}_{(n)i}$ from the previous iteration. Therefore matrices $\boldsymbol{A}_{(n)}$ and $\boldsymbol{b}_{(n)}$ have to be correctly initialized first. The initialization for $\boldsymbol{A}_{(n)}$ can be chosen as a diagonal matrix with ones on the diagonal, and $\boldsymbol{b}_{(n)}$ can be initialized as a zero vector. The estimation ends when the change in parameters of transformation matrices is small enough (about 20 iterations are sufficient) [29].

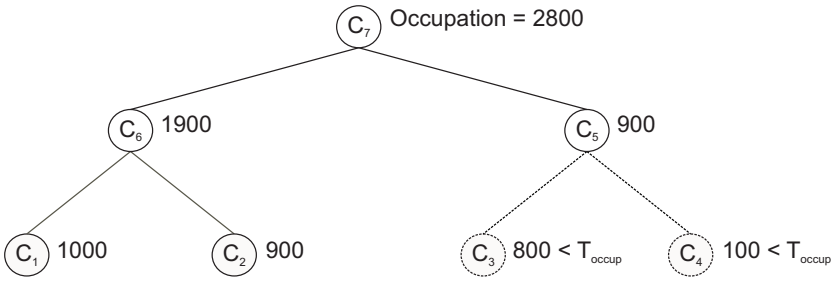

Figure 3.2: An example of a binary regression tree. Numerical values represent occupation counts of nodes (clusters). Nodes C_3 and C_4 have occupations lesser than the occupation threshold $T_{\text{occup}} = 850$, therefore all the parameters of a Gaussian that belong to C_3 and C_4 will share the transformation matrix defined for node C_5.

3.5 Regression Classes for MLLR

The advantage in MLLR-like methods is the possibility to cluster similar Gaussian parameters of the model using binary Regression Trees (RTs), where the final number of clusters depends on the amount of adaption data. All parameters belonging to the same cluster are then transformed by the same transformation. Several methods for construction of RTs were already developed, e.g. divisive hierarchical algorithms utilizing Euclidean distance between Gaussian mixture means [31], optimal clustering techniques trying to maximize the likelihood of the adaptation data [32] and others. The final set of clusters in the RT is established during the adaptation process according to the amount of data that align to Gaussians in each of the clusters – *cluster ocuppations*, where an empirical threshold has to be set. For example, let us analyze the tree depicted in Figure 3.2 with four leaves $\{C_1, C_2, C_3, C_4\}$. The clusters C_3 and C_4 have small occupation counts (lower than the threshold $T_{\text{occup}} = 850$), therefore all components in clusters C_3 and C_4 will be transformed with matrices defined for the cluster C_5. On the other hand, two individual transformation matrices will be used for C_1 and C_2.

3.6 Conclusion and Remarks

Since the adaptation process can be divided into two stages (as mentioned in Section 3.1), there is no need to specify the adaptation type in advance. It can be defined after the adaptation data have been accumulated. The computational costs are also low since the updating matrices $\boldsymbol{G}_{(n)i}, \boldsymbol{k}_{(n)i}$ do not have to be recalculated in each iteration (they are computed only once at the end of the adaptation). It also turns out to be useful to combine some of the adaptaion techniques. A suitable combination is that of MAP and fMLLR [33] – in the order MAP after fMLLR. The fMLLR method transforms all the Gaussians from the same cluster at once, thus Gaussian with insufficient amount of adaptation data are transformed as well. MAP affects each of the Gaussians separately, however only Gaussians with sufficient amount of data are improved (shifted towards adaptation data). Hence, the second MAP-pass can be thought of as a refinement stage of Gaussians with sufficient amount of data, whereas all the other Gaussians are adjusted only by fMLLR.

Another important fact to recall is that MLLR and fMLLR are based on the ML estimation – none prior information about the model parameters (transformation matrices $\boldsymbol{W}_{(n)}$) is included. Hence, when the amount of data is too low MLLR/fMLLR can completely spoil the estimates of model parameters. Several approaches were already proposed. They are based e.g. on proper initialization of the accumulators $\boldsymbol{G}_{(n)i}$ and $\boldsymbol{k}_{(n)i}$ by statistics estimated on a larger population of speakers [34]. Thus, $\boldsymbol{W}_{(n)}$ is pulled to an identity matrix [35]. Another approach is based on an additional reduction

of number of free parameters of the transformation matrices assuming a low dimensional basis [36].

It should be stated, that the transformation matrices $\boldsymbol{W}_{(n)}$ carry information about the speakers identity. More precisely they give us advice how should be the SI model "moved" to fit a new speaker, and, of course, this information should differ for each speaker (under the assumption that the initial SI model remains the same).

Chapter 4

Kernels & Mappings

In this chapter we will discuss in more depth the problem of high dimensional mappings of feature vectors, in the task of speaker recognition often denoted as *supervectors*, and both linear and non-linear kernels outlined in Section 2.3. The majority of presented mappings will be related to generative models, more precisely to GMMs. First connection between generative models and supervectors was made by Jaakkola and Haussler [37]. They proposed a method how to map a sequence of feature vectors varying in length to a vector of high, but fixed dimension utilizing generative models. Experiments were performed in the context of DNA and protein sequence analysis. During a short period of time the approach was extended also to the task of speaker recognition by people working in the IBM Thomas J. Watson Research Center, NY [38], and further investigations were carried out by Wan [39], Smith and Gales [40] and others. These mappings are based on GMM derivatives (see Section 4.4) and they were used along with kernels (mainly linear) and Support Vector Machines (SVMs) as a discriminative trainer. Since both generative and discriminative trainers were included in the estimation process, the concept is also known as *Hybrid Modeling*. The main difference between generative and discriminative methods is that generative algorithms process the input data belonging to separate classes individually and focus on efficient description of submitted data, whereas discriminative algorithms consider and process the whole set at once seeking for boundaries between different classes. Models incorporating both techniques are able to generate new data samples, and at the same time they make use of the discriminative power of discriminative classifiers. With time new mappings and new kernels arose, which are going to be discussed in following sections.

4.1 Basic Structure

We will adhere to a basic hierarchical structure of kernels proposed in [41], where *parametric* and *derivative* kernels are united under the concept of *dynamic kernel* (also known as *sequence kernel*). Parametric kernels utilize *parametric mappings*, where the name *parametric* points out that supervectors will be composed directly from parameters of some statistical models, whereas the *derivative mappings* used in derivative kernels will utilize some additional function/operator in order to extract a "derived" information from these statistical models. A block diagram of the training procedure of a target speaker model is shown in Figure 4.1.

Linear dynamic kernels have the form

$$K(\boldsymbol{O}^s, \boldsymbol{O}^k, \boldsymbol{\theta}) = \psi(\boldsymbol{O}^s; \boldsymbol{\theta})^{\mathrm{T}} \psi(\boldsymbol{O}^k; \boldsymbol{\theta}), \tag{4.1}$$

where $\boldsymbol{O}^s = \{\boldsymbol{o}_1^s, \boldsymbol{o}_2^s, \ldots, \boldsymbol{o}_T^s\}, s = 1, \ldots, S$, represents the sequence of feature vectors extracted from the speech of the s^{th} speaker, $\psi(\boldsymbol{O}^s; \boldsymbol{\theta})$ stands for a mapping, which transforms the utterance to a

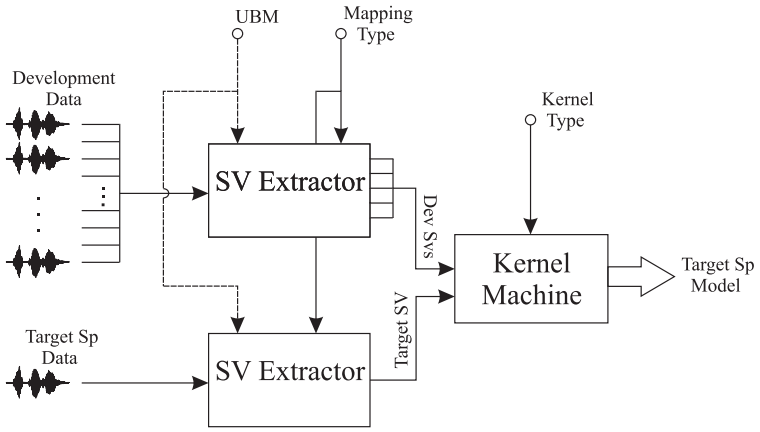

Figure 4.1: Block diagram depicting a training procedure of a target speaker model. Supervector (SV) Extractor takes as input feature vectors (depending on the utilized mapping also UBM may be required) and outputs supervectors (SVs). The target speaker's SV and SVs obtained from speakers from a development set are then redirected to the output of the kernel machine (e.g. SVM) and a proper kernel type is chosen. Output of the kernel machine is the model of the target speaker.

feature vector of a higher fixed dimension, and $\boldsymbol{\theta}$ denotes a set of parameters upon the mapping $\psi(\cdot)$ depends. Thus, the kernel can be seen as a distance metric of two vectors in feature space (generated by the mapping $\psi(\cdot)$), generally non-Euclidean. To correctly compute the dot product in such a feature space a normalization matrix $\boldsymbol{G}$ has to be introduced, which is the inverse covariance matrix of the transformed data. The kernel can be now rewritten to the form

$$K(\boldsymbol{O}^s, \boldsymbol{O}^k, \boldsymbol{\theta}) = \psi(\boldsymbol{O}^s; \boldsymbol{\theta})^{\mathrm{T}} \boldsymbol{G}\, \psi(\boldsymbol{O}^k; \boldsymbol{\theta}), \tag{4.2}$$

where

$$\boldsymbol{G}^{-1} = \mathrm{E}[(\psi(\boldsymbol{O}; \boldsymbol{\theta}) - \boldsymbol{\mu}_\psi)^{\mathrm{T}} (\psi(\boldsymbol{O}; \boldsymbol{\theta}) - \boldsymbol{\mu}_\psi)], \tag{4.3}$$

$$\boldsymbol{\mu}_\psi = \mathrm{E}[\psi(\boldsymbol{O}; \boldsymbol{\theta})], \tag{4.4}$$

and $\mathrm{E}[\cdot]$ represents the statistical expectation. In the following text the kernel function will be used in the truncated form $K(\boldsymbol{O}^s, \boldsymbol{O}^k)$, the content of $\boldsymbol{\theta}$ will be clear from the context. Note that unless stated otherwise, the notation $f(\cdot)$ will be used to distinguish functions and ordinary variables.

4.2 Parametric Mappings

Recent parametric mappings focus mainly on GMM means and/or adaptation matrices derived from Speaker Independent (SI) models according to the speaker dependent data (see Chapter 3). A very common mapping is based on the concatenation of GMM means so that a high-dimensional supervector is formed. Thus, the mapping $\psi(\cdot)$ has the form

$$\psi(\boldsymbol{O}; \boldsymbol{\lambda}) = [\boldsymbol{\mu}_1^{\mathrm{T}}, \ldots, \boldsymbol{\mu}_i^{\mathrm{T}}, \ldots, \boldsymbol{\mu}_M^{\mathrm{T}}]^{\mathrm{T}}, \tag{4.5}$$

where $\boldsymbol{\mu}_m$ are the MAP-adapted GMM means (Gaussian weights and covariance matrices remain the same), M is the number of Gaussians in the Universal Background Model (UBM – for further details see Section 3), and $\boldsymbol{\lambda} = \{\omega_m, \boldsymbol{\mu}_m, \boldsymbol{C}_m\}_{m=1}^{M}$ is the set of parameters of the generative GMM (see Section 2.2.1). To ensure the correspondence between two Gaussians of two different GMMs and the same dimensionality of subsequently formed supervectors, it is appropriate to MAP-adapt GMMs of speakers utilizing an UBM. The adaptation ensures same initial conditions in the training process and defines connections between Gaussians in the adapted GMM and the original UBM. Thus, it

is easy to determine which two Gaussians should be compared when considering two GMMs (both adapted from the same UBM).

A slightly different mapping proposed in [42] takes into consideration the difference between means of the model of a new speaker and UBM means, and can be written as

$$\begin{aligned}\psi_{\text{diff}}(\boldsymbol{O};\{\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}}\}) &= \psi(\boldsymbol{O};\boldsymbol{\lambda}) - \psi(\,\cdot\,;\hat{\boldsymbol{\lambda}}) = \\ &= [(\boldsymbol{\mu}_1 - \hat{\boldsymbol{\mu}}_1)^{\mathrm{T}},\ldots,(\boldsymbol{\mu}_i - \hat{\boldsymbol{\mu}}_i)^{\mathrm{T}},\ldots,(\boldsymbol{\mu}_M - \hat{\boldsymbol{\mu}}_M)^{\mathrm{T}}]^{\mathrm{T}}\,,\end{aligned} \tag{4.6}$$

where $\hat{\boldsymbol{\mu}}_i$ represents the mean of another fixed GMM (same for all sequences $\boldsymbol{O}^s$ related to specific speakers), and the notation $\psi(\,\cdot\,;\hat{\boldsymbol{\lambda}})$ represents the mapping of its means. When $\hat{\boldsymbol{\mu}}_i$ are taken from UBM the mapping (4.6) can be seen as the centered version of (4.5).

Another kind of supervector extraction utilizes the MLLR transformation matrices computed according to Section 3.3 and given in equation (3.13). The rows of such transformation matrices are subsequently concatenated into one high-dimensional supervector. The mapping $\psi(\cdot)$ can be expressed as

$$\psi(\boldsymbol{O};\{\boldsymbol{W}_{(k)}\}_{k=1}^{K}) = [\boldsymbol{w}_{(1)1}^{\mathrm{T}},\ldots,\boldsymbol{w}_{(1)D}^{\mathrm{T}},\boldsymbol{w}_{(2)1}^{\mathrm{T}},\ldots,\boldsymbol{w}_{(2)D}^{\mathrm{T}},\ldots,\boldsymbol{w}_{(K)1}^{\mathrm{T}},\ldots,\boldsymbol{w}_{(K)D}^{\mathrm{T}}]^{\mathrm{T}}\,, \tag{4.7}$$

where D is the dimension of feature vectors, K is the number of transformation matrices and $\boldsymbol{w}_{(k)i}^{\mathrm{T}}$ denotes the i^{th} transposed row of the matrix $\boldsymbol{W}_{(k)}$. The matrix $\boldsymbol{W}_{(k)}$ can be thought of as a "data error" from which the generative model suffers, or as the scale and direction (in the sense of a matrix transform) in which the data are located given an initial UBM. Assuming one initial UBM same for all the speakers, and further assuming that the position of their feature vectors in the acoustic space is unique, the adaptation matrix $\boldsymbol{W}_{(k)}$ should be for each speaker unique too.

A high dimensional mapping not directly connected to generative models and GMMs was proposed in [43]. It is called Generalized Linear Discriminant Sequence (GLDS), and it is based on a vector function that transforms directly the feature vectors. The mapping has the form

$$\psi(\boldsymbol{O};\boldsymbol{\varphi}(\cdot)) = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\varphi}(\boldsymbol{o}_t)\,, \tag{4.8}$$

$$\boldsymbol{\varphi}(\boldsymbol{o}_t) = [\varphi_1(\boldsymbol{o}_t),\ldots,\varphi_j(\boldsymbol{o}_t),\ldots,\varphi_J(\boldsymbol{o}_t)]^{\mathrm{T}}\,, \tag{4.9}$$

$\boldsymbol{\varphi}(\cdot)$ represents an expansion of the input space into a vector of scalar functions, $\varphi_j : R^D \mapsto R$, D is the dimension of the input space and J is the dimension of the vector function $\boldsymbol{\varphi}(\cdot)$. In [43] the considered feature expansion consisted of monomials up to the k^{th} order, e.g. for monomials up to the second order of the feature vector $\boldsymbol{o} = [o_1, o_2, \ldots, o_D]^{\mathrm{T}}$ we get

$$\boldsymbol{\varphi}(\boldsymbol{o}; k=2) = [1, o_1, \ldots, o_D, o_1^2, o_1o_2, \ldots, o_1o_D, o_2^2, o_2o_3, \ldots, o_2o_D, o_3^2, \ldots, o_D^2]\,, \tag{4.10}$$

where $\dim(\boldsymbol{o}) = D$ and $\dim(\boldsymbol{\varphi}(\boldsymbol{o};k)) = [(D+k)!]/[D!\,k!\,]$. After substituting (4.10) into (4.8) one can notice, that the mapping (4.8) comprises first- and second-order moments – the mean and the covariance of dimensions of feature vectors (higher moments will be obtained for higher order monomials) [44]. Thus, again data statistics are collected (as in the case of GMM supervectors), however now also statistics of higher order may be acquired. Further generalizations of GLDS and associated kernels allowing not only polynomial degrees but also infinite dimensional expansions were studied in [45, 46].

In [47] the vector function $\boldsymbol{\varphi}(\boldsymbol{o})$ was chosen as

$$\boldsymbol{\varphi}_{\text{PS}}(\boldsymbol{o}_t;\boldsymbol{\lambda}) = [\gamma_1(\boldsymbol{o}_t),\ldots,\gamma_m(\boldsymbol{o}_t),\ldots,\gamma_M(\boldsymbol{o}_t)]\,, \tag{4.11}$$

where $\gamma_m(\boldsymbol{o}_t)$ is the m^{th} Gaussian's posterior specified in (3.4). The mapping based on $\boldsymbol{\varphi}_{\text{PS}}(\boldsymbol{o_t};\boldsymbol{\lambda})$ is called the Probabilistic Sequence (PS) mapping and has the same form as (4.8) except for the vector function, hence

$$\psi(\boldsymbol{O};\boldsymbol{\varphi}_{\text{PS}}(\cdot)) = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\varphi}_{\text{PS}}(\boldsymbol{o}_t)\,. \tag{4.12}$$

In order to use such a mapping, it is useful to cover the part of the acoustic space of interest with well distributed Gaussians over the area, e.g. it is useful to utilize some anchor models chosen from the background population of speakers [44].

4.3 Parametric Kernels

4.3.1 Mean Supervector Based Kernels

A basic, frequently used and well-working kernel is the one in (4.2), where the matrix $\boldsymbol{G}$ is assumed nearly always diagonal (as its size is often huge and there are not enough data for its full estimation, e.g. to ensure its invertibility) – in many cases even replaced by the identity matrix [39].

Other kernels utilize mainly the Kullback-Leibler Divergence (KLD), which is a standard tool used when comparing two GMMs. Since KLD cannot be expressed in closed form solution when multi-Gaussian GMMs are involved several approximations have to be made. First KLD kernel was proposed in [48], where the unapproximated KLD was utilized using a single Gaussian model. This kernel was denoted as Probabilistic Distance Kernel (PDK). Further investigations were carried out in [49, 50]. In [50] two kernels were proposed – Supervector Linear Kernel (SLK) and L^2 Inner Product Kernel (L^2IPK). The SLK is an extension of the PDK, and it is based on the approximated Kullback-Leibler Divergence (KLD). In [51] not only the GMM means, but additionally also the GMM weights were adapted and used when constructing the kernel function. All of these kernels are going to be reviewed now.

Probabilistic Distance Kernels (PDK) was proposed in [48] and as already mentioned, they utilize the KLD. The symmetricity of such a dissimilarity measure was preserved using symmetric KLD of the form

$$D_{\text{S}}(p(\boldsymbol{o}|\boldsymbol{\lambda}_1)\,||\,p(\boldsymbol{o}|\boldsymbol{\lambda}_2)) = \int_{-\infty}^{\infty} p(\boldsymbol{o}|\boldsymbol{\lambda}_1)\log\left(\frac{p(\boldsymbol{o}|\boldsymbol{\lambda}_1)}{p(\boldsymbol{o}|\boldsymbol{\lambda}_2)}\right)\text{d}\boldsymbol{o} + \int_{-\infty}^{\infty} p(\boldsymbol{o}|\boldsymbol{\lambda}_2)\log\left(\frac{p(\boldsymbol{o}|\boldsymbol{\lambda}_2)}{p(\boldsymbol{o}|\boldsymbol{\lambda}_1)}\right)\text{d}\boldsymbol{o}. \tag{4.13}$$

and the validity of the kernel was ensured introducing equation

$$K_{\text{PDK}}(\boldsymbol{O}^{spk1},\boldsymbol{O}^{spk2}) = \text{e}^{-a_1 D_{\text{S}}(p(\boldsymbol{o}|\boldsymbol{\lambda}_1)\,||\,p(\boldsymbol{o}|\boldsymbol{\lambda}_2))+a_2}\,, \tag{4.14}$$

where a_1, a_2 represent scale and shift factors, respectively. They were involved because of stability reasons and were set empirically. In the case of GMM there is no analytic solution of (4.13) and some numerical approximations have to be employed (e.g. [52]), but in the case of a full covariance GMM with a single Gaussian, the distance in (4.13) can be computed directly as

$$\begin{aligned} D_{\text{S}}(\mathcal{N}(\cdot;\boldsymbol{\mu}_1,\boldsymbol{C}_1)\,||\,\mathcal{N}(\cdot;\boldsymbol{\mu}_2,\boldsymbol{C}_2)) = \frac{1}{2}\big(&\text{tr}\left(\boldsymbol{C}_1\boldsymbol{C}_2^{-1}\right) + \text{tr}\left(\boldsymbol{C}_1^{-1}\boldsymbol{C}_2\right) - 2I + \\ &+ \text{tr}\left[\left(\boldsymbol{C}_1^{-1}+\boldsymbol{C}_2^{-1}\right)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^{\text{T}}\right]\big)\,, \end{aligned} \tag{4.15}$$

where $I = \dim(\boldsymbol{o})$. Thus, not only mean vectors are involved, but also covariance matrices (however just for single Gaussian models). It is easy to see, that for $\boldsymbol{C} = \boldsymbol{C}_1 = \boldsymbol{C}_2$, the distance (4.15)

degenerates to

$$D_{\mathrm{S}}(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{C}) \,||\, \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{C})) = d(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2; \boldsymbol{C}) = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^{\mathrm{T}} \boldsymbol{C}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\,. \tag{4.16}$$

Further extensions to multi-Gaussian GMMs were performed using the KLD approximation [52] via log sum inequality. The approximation may be expressed as

$$D\left(\sum_{m=1}^{M} \omega_m \mathcal{N}_m \,||\, \sum_{m=1}^{M} \tilde{\omega}_m \tilde{\mathcal{N}}_m\right) \leq \sum_{m=1}^{M} D(\omega_m \,||\, \tilde{\omega}_m) + \sum_{m=1}^{M} \omega_m D(\mathcal{N}_m \,||\, \tilde{\mathcal{N}}_m), \tag{4.17}$$

where $D(\mathcal{N}_m \,||\, \tilde{\mathcal{N}}_m)$, $D(\omega_m \,||\, \tilde{\omega}_m)$ is the KLD between the m^{th} Gaussian and m^{th} Gaussian's weight of the model $\boldsymbol{\lambda}$ and the model $\tilde{\boldsymbol{\lambda}}$, respectively. The question, which two Gaussians should be "compared" against each other when evaluating $D(\mathcal{N}_m \,||\, \tilde{\mathcal{N}}_m)$ and $D(\omega_m \,||\, \tilde{\omega}_m)$ is implicitly solved by the MAP adaptation of UBM when estimating parameters $\boldsymbol{\lambda}$, $\tilde{\boldsymbol{\lambda}}$ of both GMMs (see discussion in Section 4.2). Sufficient condition for left- and right-hand side in (4.17) to equal is $\gamma_m(t) = \tilde{\gamma}_m(t)$ for all pairs $m = 1, \ldots, M$, where the posterior $\gamma_m(t)$ is given in (3.4).

Assuming equal weights (thus $D(\omega_m \,||\, \tilde{\omega}_m) = 0$) and covariances in both models, and symmetric KLD $D_{\mathrm{S}}(\boldsymbol{\lambda}||\tilde{\boldsymbol{\lambda}}) = D(\boldsymbol{\lambda}||\tilde{\boldsymbol{\lambda}}) + D(\tilde{\boldsymbol{\lambda}}||\boldsymbol{\lambda})$ we get

$$D_{\mathrm{S}}(\boldsymbol{\lambda}||\tilde{\boldsymbol{\lambda}}) \leq \sum_{m=1}^{M} \omega_m d(\boldsymbol{\mu}, \tilde{\boldsymbol{\mu}}; \boldsymbol{C}) = \sum_{m=1}^{M} \omega_m (\boldsymbol{\mu}_m - \tilde{\boldsymbol{\mu}}_m)^{\mathrm{T}} \boldsymbol{C}_m^{-1} (\boldsymbol{\mu}_m - \tilde{\boldsymbol{\mu}}_m), \tag{4.18}$$

however as stated above this is not a valid kernel (e.g. it cannot be decomposed to a dot product of the form $K = \phi(\boldsymbol{x})^{\mathrm{T}} \phi(\boldsymbol{x})$, where $\phi(\cdot)$ is some mapping function of $\boldsymbol{x}$).

Supervector Linear Kernel (SLK) follows the conclusions made in the PDK, but uses the KLD approximation (4.18). It is adjusted in order to ensure the validity of the Mercer's condition (2.38) to get a valid kernel. Only the inner product between distinct but adjacent means is taken. Assuming diagonal covariances the resulting kernel can be written in the form

$$K_{\mathrm{SLK}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \sum_{m=1}^{M} \omega_m (\boldsymbol{\mu}_m^{spk1})^{\mathrm{T}} \boldsymbol{C}_m^{-1} \boldsymbol{\mu}_m^{spk2} = \psi(\boldsymbol{O}^{spk1}; \boldsymbol{\lambda})^{\mathrm{T}}\, \boldsymbol{\Omega}\tilde{\boldsymbol{G}}\, \psi(\boldsymbol{O}^{spk2}; \boldsymbol{\lambda})\,, \tag{4.19}$$

$$\tilde{\boldsymbol{G}} : \mathrm{diag}(\tilde{\boldsymbol{G}}) = [\mathrm{diag}(\boldsymbol{C}_1^{-1}), \ldots, \mathrm{diag}(\boldsymbol{C}_m^{-1}), \ldots, \mathrm{diag}(\boldsymbol{C}_M^{-1})]^{\mathrm{T}}\,, \tag{4.20}$$

$$\boldsymbol{\Omega} : \mathrm{diag}(\boldsymbol{\Omega}) = [\omega_1, \ldots, \omega_M]^{\mathrm{T}} \otimes \boldsymbol{1}_D\,, \tag{4.21}$$

where $\boldsymbol{\lambda} = \{\omega_m, \boldsymbol{\mu}_m, \boldsymbol{C}_m\}_{m=1}^{M}$ are parameters of a GMM given in (2.16), $\boldsymbol{\mu}_m^{spk1}, \boldsymbol{\mu}_m^{spk2}$ are adjacent, speaker dependent, MAP-adapted means, $\boldsymbol{1}_D$ is a D dimensional vector of ones, where $D = \dim(\boldsymbol{o}) = \dim(\boldsymbol{\mu})$, $\otimes$ denotes the Kronecker product, the function $\mathrm{diag}(\boldsymbol{X})$ transforms a matrix $\boldsymbol{X}$ to a vector with its entries equal to the diagonal of the matrix $\boldsymbol{X}$, and $\tilde{\boldsymbol{G}}$, $\boldsymbol{\Omega}$ are zero matrices with diagonals specified in (4.20), (4.21), respectively. The matrix $\boldsymbol{C}_m$ can be taken directly from the UBM as the UBM's covariances were not adapted. The matrix $\tilde{\boldsymbol{G}}$ can be thought of as an approximation of (4.3) and $\boldsymbol{\Omega}$ represents an additional weighting factor. There are several advantages of such a kernel [43]. The matrix $\boldsymbol{R} = \boldsymbol{\Omega}\tilde{\boldsymbol{G}}$ can be factored using Cholesky decomposition yielding $\boldsymbol{R} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}$ (in fact $\boldsymbol{R}$ is diagonal, thus $\boldsymbol{U}$ contains square roots of $\boldsymbol{R}$ on its diagonal), and all the supervectors can be transformed before the SVM training as $\boldsymbol{y} = \boldsymbol{U}\psi(\boldsymbol{O}; \boldsymbol{\lambda})$. The kernel takes the form

$$K_{\mathrm{SLK}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \boldsymbol{y}_{spk1}^{\mathrm{T}} \boldsymbol{y}_{spk2}\,, \tag{4.22}$$

there is no need to multiply supervectors with $\boldsymbol{R}$ whenever the kernel function is evaluated and the computational costs during the training will be reduced. Furthermore, the model compaction

technique can be applied, thus the SVM decision hyperplane (2.35) with L support vectors can be computed in advance and the final decision function can be expressed as

$$f(\boldsymbol{O}^j) = \left(\boldsymbol{\Omega}\tilde{\boldsymbol{G}}\sum_{k=1}^{L}\alpha_k y_k \psi(\boldsymbol{O}^k;\boldsymbol{\lambda})\right)^{\mathrm{T}} \psi(\boldsymbol{O}^j;\boldsymbol{\lambda}) + b\,. \tag{4.23}$$

L^2 Inner Product Kernel (L^2IPK) is derived from the function space inner product of two GMMs and has the form

$$K_{\mathrm{L}^2\mathrm{IPK}}(\boldsymbol{O}^{spk1},\boldsymbol{O}^{spk2}) = \int_{R^n} g_{spk1}(\boldsymbol{o}) g_{spk2}(\boldsymbol{o})\,\mathrm{d}\boldsymbol{o}. \tag{4.24}$$

The closed form solution has the form

$$K_{\mathrm{L}^2\mathrm{IPK}}(\boldsymbol{O}^{spk1},\boldsymbol{O}^{spk2}) = \sum_{i=1}^{M}\sum_{j=1}^{M}\omega_i\omega_j\mathcal{N}\left(\boldsymbol{\mu}_i^{spk1}-\boldsymbol{\mu}_j^{spk2};\boldsymbol{0},\boldsymbol{C}_i+\boldsymbol{C}_j\right), \tag{4.25}$$

where $\boldsymbol{0}$ stands for a zero vector. Under the assumption that means corresponding to the i^{th} and j^{th} Gaussian for $i \neq j$ lie far apart, (4.25) can be simplified to the form

$$K_{\mathrm{L}^2\mathrm{IPK}}(\boldsymbol{O}^{spk1},\boldsymbol{O}^{spk2}) = \sum_{i=1}^{N}\sum_{j=1}^{N}\omega_i^2\mathcal{N}(\boldsymbol{\mu}_i^{spk1}-\boldsymbol{\mu}_j^{spk2};\boldsymbol{0},2\boldsymbol{C}_i). \tag{4.26}$$

PDK2 was proposed in [49]. It is based on a similar approach to SLK, however the Mercer's condition is guaranteed by (4.14) with $a_1 = 1, a_2 = 0$ rather than taking only the inner product between distinct but adjacent means. The kernel can be expressed as

$$\begin{aligned} K_{\mathrm{PDK2}}(\boldsymbol{O}^{spk1},\boldsymbol{O}^{spk2}) &= \mathrm{e}^{-\sum_{m=1}^{M}\omega_m d(\boldsymbol{\mu}_m^{spk1},\boldsymbol{\mu}_m^{spk2};\boldsymbol{C}_m)} \\ &= \mathrm{e}^{-\sum_{m=1}^{M}\omega_m(\boldsymbol{\mu}_m^{spk1}-\boldsymbol{\mu}_m^{spk2})^{\mathrm{T}}\boldsymbol{\Omega}\tilde{\boldsymbol{G}}(\boldsymbol{\mu}_m^{spk1}-\boldsymbol{\mu}_m^{spk2})} \\ &= \mathrm{e}^{-\left\|\boldsymbol{U}(\psi(\boldsymbol{O}^{spk1};\boldsymbol{\lambda})-\psi(\boldsymbol{O}^{spk2};\boldsymbol{\lambda}))\right\|^2}, \end{aligned} \tag{4.27}$$

where $\boldsymbol{R} = \boldsymbol{\Omega}\tilde{\boldsymbol{G}} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}$, $\boldsymbol{\Omega}$ and $\tilde{\boldsymbol{G}}$ were specified in (4.21) and (4.20), respectively.

4.3.2 One-class MLLR Kernels

In this section we will focus only on supervectors constructed from one transformation matrix (global transformation) common for all the model means. Loosely speaking, only one cluster is constructed containing all the model parameters (in this case the GMM means), thus only one transformation matrix is computed for each speaker.

The kernel function considered can be a simple linear inner product (4.1) of MLLR based supervectors as used in [53], nevertheless one could also transform the means according to the formula (3.12) and utilize kernels described in Subsection 4.3.1.

A more flexible kernel approach was proposed in [54]. It arises from (4.19), where means $\boldsymbol{\mu}_m$ are transformed according to equation (3.12), hence

$$K(\boldsymbol{O}^{spk1},\boldsymbol{O}^{spk2}) = \sum_{m=1}^{M}\left(\boldsymbol{\Delta}_m^{\frac{1}{2}}(\boldsymbol{A}\boldsymbol{\mu}_m+\boldsymbol{b})\right)^{\mathrm{T}}\left(\boldsymbol{\Delta}_m^{\frac{1}{2}}(\boldsymbol{H}\boldsymbol{\mu}_m+\boldsymbol{d})\right), \tag{4.28}$$

where $[\boldsymbol{A}, \boldsymbol{b}]$ and $[\boldsymbol{H}, \boldsymbol{d}]$ are global transformation matrices for speakers $spk1$ and $spk2$, respectively, and $\boldsymbol{\Delta}_m = \omega_m \boldsymbol{C}_m^{-1}$. After expanding equation (4.28) we get

$$\begin{aligned} K(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = &\sum_{m=1}^{M} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{A}\boldsymbol{\mu}_m\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{H}\boldsymbol{\mu}_m\right) \quad + \sum_{m=1}^{M} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{A}\boldsymbol{\mu}_m\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{d}\right) + \\ &+ \sum_{m=1}^{M} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{b}\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{H}\boldsymbol{\mu}_m\right) \qquad + \sum_{m=1}^{M} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{b}\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{d}\right) . \end{aligned} \tag{4.29}$$

Let us have a look on the first term in (4.29). Some notations shall be introduced, $\mathrm{tr}(\boldsymbol{A})$ stands for the trace of the matrix $\boldsymbol{A}$, $\boldsymbol{e}_k$ is a zero vector with 1 on its k^{th} position, Δ_{ik} is the k^{th} diagonal element of the matrix $\boldsymbol{\Delta}_i$, D represents the number of rows in $\boldsymbol{A}$ and $\boldsymbol{a}_k$ equals the transpose of the k^{th} row of $\boldsymbol{A}$. Then (assuming diagonal covariance matrices $\boldsymbol{C}_m$)

$$\begin{aligned} \sum_{m=1}^{M} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{A}\boldsymbol{\mu}_m\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{H}\boldsymbol{\mu}_m\right) &= \sum_{m=1}^{M} \mathrm{tr}\left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{A}\boldsymbol{\mu}_m\boldsymbol{\mu}_m^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{\Delta}_m^{\frac{1}{2}}\right) = \sum_{m=1}^{M} \mathrm{tr}\left(\boldsymbol{\Delta}_m \boldsymbol{A}\boldsymbol{\mu}_m\boldsymbol{\mu}_m^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\right) = \\ &= \sum_{m=1}^{M} \mathrm{tr}\left[\left(\sum_{k=1}^{D} \Delta_{mk} \boldsymbol{e}_k \boldsymbol{e}_k^{\mathrm{T}}\right) \boldsymbol{A}\boldsymbol{\mu}_m\boldsymbol{\mu}_m^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\right] = \\ &= \sum_{m=1}^{M} \sum_{k=1}^{D} \mathrm{tr}\left[\boldsymbol{e}_k \boldsymbol{e}_k^{\mathrm{T}} \boldsymbol{A} \left(\Delta_{mk} \boldsymbol{\mu}_m \boldsymbol{\mu}_m^{\mathrm{T}}\right) \boldsymbol{H}^{\mathrm{T}}\right] = \\ &= \sum_{k=1}^{D} \mathrm{tr}\left[\boldsymbol{e}_k^{\mathrm{T}} \boldsymbol{A} \left(\sum_{m=1}^{M} \Delta_{mk} \boldsymbol{\mu}_m \boldsymbol{\mu}_m^{\mathrm{T}}\right) \boldsymbol{H}^{\mathrm{T}} \boldsymbol{e}_k\right] = \\ &= \sum_{k=1}^{D} \boldsymbol{a}_k^{\mathrm{T}} \left(\sum_{m=1}^{M} \Delta_{mk} \boldsymbol{\mu}_m \boldsymbol{\mu}_m^{\mathrm{T}}\right) \boldsymbol{h}_k = \\ &= \sum_{k=1}^{D} \boldsymbol{a}_k^{\mathrm{T}} \boldsymbol{R}_k \boldsymbol{h}_k . \end{aligned} \tag{4.30}$$

All the other terms in (4.29) can be rearranged in a similar fashion, what results in

$$\sum_{m=1}^{M} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{A}\boldsymbol{\mu}_m\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{d}\right) = \sum_{k=1}^{D} d_k \boldsymbol{a}_k^{\mathrm{T}} \boldsymbol{r}_k , \tag{4.31}$$

$$\sum_{m=1}^{M} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{b}\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{H}\boldsymbol{\mu}_m\right) = \sum_{k=1}^{D} b_k \boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{h}_k , \tag{4.32}$$

$$\sum_{m=1}^{M} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{b}\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} \boldsymbol{d}\right) = \sum_{k=1}^{D} b_k d_k \delta_k , \tag{4.33}$$

where $\boldsymbol{r}_k = \sum_{m=1}^{M} \Delta_{mk} \boldsymbol{\mu}_m$ and $\delta_k = \sum_{m=1}^{M} \Delta_{mk}$. Now, the kernel (4.28) can be rewritten as

$$\begin{aligned} K(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) &= \sum_{k=1}^{D} \boldsymbol{a}_k^{\mathrm{T}} \boldsymbol{R}_k \boldsymbol{h}_k + d_k \boldsymbol{a}_k^{\mathrm{T}} \boldsymbol{r}_k + b_k \boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{h}_k + b_k d_k \delta_k = \\ &= \psi(\boldsymbol{O}^{spk1}; [\boldsymbol{A}, \boldsymbol{b}])^{\mathrm{T}} \boldsymbol{Q}\, \psi(\boldsymbol{O}^{spk2}; [\boldsymbol{H}, \boldsymbol{d}]) . \end{aligned} \tag{4.34}$$

The matrix $\boldsymbol{Q}$ is symmetric, positive-definite (the kernel satisfies the Mercer's condition (2.38)) as it arises from (4.19) and it consists of D blocks of size $(D+1) \times (D+1)$, where each block $\boldsymbol{Q}_k$ can be

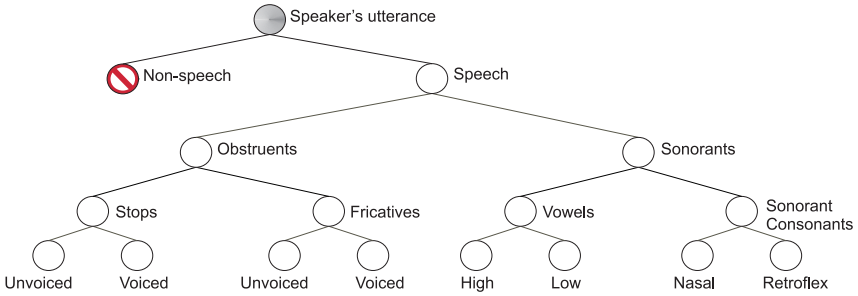

Figure 4.2: Regression tree based on phonetic classes.

expressed as

$$\boldsymbol{Q}_k = \begin{pmatrix} \boldsymbol{R}_k & \boldsymbol{r}_k \\ \boldsymbol{r}_k^{\mathrm{T}} & \delta_k \end{pmatrix} . \tag{4.35}$$

The matrix $\boldsymbol{Q}$ depends only on the UBM, therefore is the same for all speakers and can be computed in advance. Another advantage of the block diagonal property of $\boldsymbol{Q}$ is the possibility to easily compute the Cholesky decomposition of $\boldsymbol{Q}$ and apply the model compaction technique as discussed in Section 4.3.1.

Note that the dimension of MLLR based supervectors ($D\times(D+1)$; $D = \dim(\boldsymbol{o}_t)$) is in comparison with the dimension of mean-supervetors ($D \times M$) often significantly lower since the number M of Gaussians in the UBM is often high.

4.3.3 Multi-class MLLR Kernels

The extension to the multi-class case is straightforward. The regression tree is involved (see Section 3.5), thus several transformation matrices are computed at a time. The crucial problem to be solved is the construction of the regression tree. The approaches in Section 3.5 do not directly concern the dissimilarities between speakers. They mainly focus on clustering of features close in the acoustic space without an explicit knowledge how do these features characterize the speaker's identity. A method how to handle such a problem was introduced in [53], where the regression tree is designed according to broad phonetic classes depicted in Figure 4.2. Each of the classes contains a set of Gaussians for which a separate transformation matrix is going to be computed. The use of an text-independent GMM is now unfeasible – one cannot decide which Gaussian belongs to which phonetic class. The possible solution would be to utilize the Large Vocabulary Continuous Speech Recognition (LVCSR) system based on a set of Hidden Markov Models (HMMs) with states formed by GMMs, where each HMM represents an elementary linguistic unit (e.g. monophone, triphone, syllable, etc.). Thus, a decision about the phonetic class pertinence can be made. Such an LVCSR system can be adapted according to formulas in Section 3.3 and the regression tree depicted in Figure 4.2. Hence, the resulting matrices will correspond to proposed phonetic broad classes.

Another method was presented in [55]. It uses an UBM with M Gaussians containing diagonal covariances and an open-loop phonetic recognizer (phonetic recognition without lexical or phonotactic constraints). The training data are partitioned with the use of an open-loop phonetic recognizer into several clusters in order to match the phonetic classes. Let us consider a two MLLR-class case – clusters will be created for obstruents and sonorants (see Figure 4.2). The proposed UBM has the form

$$g(\boldsymbol{o}_t) = \vartheta_S \sum_{m=1}^{M/2} \omega_m \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_m, \boldsymbol{C}_m) + \vartheta_O \sum_{m=M/2+1}^{M} \omega_m \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_m, \boldsymbol{C}_m) , \tag{4.36}$$

where ϑ_S, ϑ_O are the weights of Gaussians associated to sonorants and obstruents, respectively. They are calculated as the percentage of frames assigned to particular clusters. Each of the $M/2$ Gaussians is trained using the Expectation Maximization (EM) algorithm separately (from the appropriate data cluster), and they are combined at the end to form the UBM given in (4.36). Note that UBM weights have to be rescaled to sum to unity. All three models (one UBM with M Gaussians and two models with $M/2$ Gaussians) are employed in the adaptation phase. Firstly an occupation threshold T_{occup} has to be specified as in Section 3.5. The open-loop phonetic recognizer is used again to redistribute the test utterance frames between participating phonetic classes. For classes, where the number of frames is higher than T_{occup}, the models with $M/2$ Gaussians (trained for appropriate clusters) are adapted. For classes not satisfying the criterion, the UBM with M Gaussians will be adapted. The higher MLLR-class case can be derived in analogy with before mentioned technique.

Such an approach is in principle very similar to an adaptation of the LVCSR system, but it is much easier to control the influence of Gaussians in each of the phonetic classes. Generally, the number of Gaussians associated to each class can be different from $M/2$, but the authors proclaim that the performance decreases.

The last point to discuss is which kernel to use when multiple matrices are present. The kernel function can be chosen similarly to Section 4.3.2. One can consider only a simple linear inner product of MLLR based supervectors, or transform the model according to the formula (3.12) and use kernels from Subsection 4.3.1, or extend the approach described at the end of Subsection 4.3.2 as was done in [55]. The extension is straightforward if UBMs in the form of equation (4.36) are used. Nevertheless, with some effort the method could be easily extended also to cases when the LVCSR system is in use.

Let us have a look on UBMs defined as in (4.36). The kernel function (4.28) takes now the form

$$\begin{aligned} K(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) &= \vartheta_{\mathrm{S}} K_{\mathrm{S}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) + \vartheta_{\mathrm{O}} K_{\mathrm{O}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \\ &= \vartheta_{\mathrm{S}} \sum_{m=1}^{M/2} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} (\boldsymbol{A}_{\mathrm{S}} \boldsymbol{\mu}_m + \boldsymbol{b}_{\mathrm{S}}) \right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} (\boldsymbol{H}_{\mathrm{S}} \boldsymbol{\mu}_m + \boldsymbol{d}_{\mathrm{S}}) \right) + \\ &+ \vartheta_{\mathrm{O}} \sum_{m=M/2+1}^{M} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} (\boldsymbol{A}_{\mathrm{O}} \boldsymbol{\mu}_m + \boldsymbol{b}_{\mathrm{O}}) \right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_m^{\frac{1}{2}} (\boldsymbol{H}_{\mathrm{O}} \boldsymbol{\mu}_m + \boldsymbol{d}_{\mathrm{O}}) \right) . \end{aligned} \tag{4.37}$$

Thus, the problem can be divided into two subproblems solved separately for $K_{\mathrm{S}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2})$ and $K_{\mathrm{O}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2})$. Two solutions are obtained

$$K_{\mathrm{S}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \psi_{\mathrm{S}}(\boldsymbol{O}^{spk1}; [\boldsymbol{A}_{\mathrm{S}}, \boldsymbol{b}_{\mathrm{S}}])^{\mathrm{T}} \boldsymbol{Q}_{\mathrm{S}}\, \psi_{\mathrm{S}}(\boldsymbol{O}^{spk2}; [\boldsymbol{H}_{\mathrm{S}}, \boldsymbol{d}_{\mathrm{S}}]) , \tag{4.38}$$

$$K_{\mathrm{O}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \psi_{\mathrm{O}}(\boldsymbol{O}^{spk1}; [\boldsymbol{A}_{\mathrm{O}}, \boldsymbol{b}_{\mathrm{O}}])^{\mathrm{T}} \boldsymbol{Q}_{\mathrm{O}}\, \psi_{\mathrm{O}}(\boldsymbol{O}^{spk2}; [\boldsymbol{H}_{\mathrm{O}}, \boldsymbol{d}_{\mathrm{O}}]) , \tag{4.39}$$

where $\psi_{\mathrm{S}}(\cdot)$ is the sonorant part and $\psi_{\mathrm{O}}(\cdot)$ is the obstruent part of MLLR based supervectors. The matrices $\boldsymbol{Q}_{\mathrm{S}}$, $\boldsymbol{Q}_{\mathrm{O}}$ are block diagonal with blocks defined in (4.35). The final kernel is

$$K(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \begin{bmatrix} \psi_{\mathrm{S}}(\boldsymbol{O}^{spk1})^{\mathrm{T}} & \psi_{\mathrm{O}}(\boldsymbol{O}^{spk1})^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \vartheta_{\mathrm{S}} \boldsymbol{Q}_{\mathrm{S}} & \boldsymbol{0} \\ \boldsymbol{0} & \vartheta_{\mathrm{O}} \boldsymbol{Q}_{\mathrm{O}} \end{bmatrix} \begin{bmatrix} \psi_{\mathrm{S}}(\boldsymbol{O}^{spk2}) \\ \psi_{\mathrm{O}}(\boldsymbol{O}^{spk2}) \end{bmatrix} , \tag{4.40}$$

where the conditional parts of $\psi_{\mathrm{S}}(\cdot)$ and $\psi_{\mathrm{O}}(\cdot)$ were omitted because of lucidity. Properties of $\boldsymbol{Q}$ mentioned at the end of Section 4.3.2 are preserved. It should be stated, that the two approaches, either the use of the kernel (4.40) or the use of the kernel (4.19), where the means are transformed according to the formula (3.12), are equivalent. The second method outperforms the first one in terms of computational costs as there is no need to transform each mean of the model [55] (this is useful especially when LVCSR systems are utilized).

Also note that when supervectors are constructed, some matrices may occur repeatedly in cases when two separate classes with insufficient amount of data descend from the same parent class, see Figure 3.2.

4.4 Derivative Mappings

The derivative kernels are based on the work of Jaakkola and Haussler [37], who made the first connection between generative and discriminative models at all. Further investigations were carried out by Wan [39] and by Smith and Gales [40]. They have proposed generalizations in the form of score spaces defined by the mapping

$$\psi_{\tilde{F}}^{f}(\boldsymbol{O};\{\boldsymbol{\lambda}_q\}_{q=1}^{Q}) = \psi_{\tilde{F}}(f(\{p(\boldsymbol{O}|\boldsymbol{\lambda}_q)\}_{q=1}^{Q}))\,, \tag{4.41}$$

where $\{p(\boldsymbol{O}|\boldsymbol{\lambda}_q)\}_{q=1}^{Q}$ is a set of Q generative models. The function $f(\cdot)$ is called the score argument and it determines the form of the output of the set of generative models and it is mapped by the score operator $\tilde{F}$ to a fixed-dimensional score space $\psi_{\tilde{F}}^{f}(\boldsymbol{O};\{\boldsymbol{\lambda}_q\}_{q=1}^{Q})$ [40]. The purpose of the score operator is to extract useful discriminative information from generative models – for derivative score spaces derivative operators are considered such as the zeroth-order, first-order and higher-order derivatives with respect to the parameters of the generative model. Nevertheless, because of computational complexity of higher-order derivatives mainly the zeroth and first-order derivatives are studied. In this work only the case to the limit $Q = 2$ and score argument based on logarithmic function will be discussed. A very detailed description of the whole problem can be found in [10]. The score space (4.41) for $Q = 1$, known as Log-Likelihood Score (LLS) space, can be expressed in the form

$$\psi_{\nabla}^{\mathrm{LLS}}(\boldsymbol{O};\hat{\boldsymbol{\lambda}},\rho) = \frac{1}{T}\left[\nabla_{\boldsymbol{\lambda}}^{(0,\rho)} \ln p(\boldsymbol{O}|\boldsymbol{\lambda})\Big|_{\hat{\boldsymbol{\lambda}}}\right] = \frac{1}{T}\begin{bmatrix} \ln p(\boldsymbol{O}|\hat{\boldsymbol{\lambda}}) \\ \nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{O}|\boldsymbol{\lambda})\Big|_{\hat{\boldsymbol{\lambda}}} \\ \vdots \\ \mathrm{vec}\left(\nabla_{\boldsymbol{\lambda}}^{\rho} \ln p(\boldsymbol{O}|\boldsymbol{\lambda})\Big|_{\hat{\boldsymbol{\lambda}}}\right) \end{bmatrix}, \tag{4.42}$$

where the term $\frac{1}{T}$ was introduced because of the sequence length normalization with T equal to the number of feature vectors in $\boldsymbol{O} = \{\boldsymbol{o}_t\}_{t=1}^{T}$, the function $\mathrm{vec}(\cdot)$ transforms a matrix into a column vector and ρ defines the order of the derivative. The score space for $Q = 2$, known as Log-Likelihood Ratio Score (LLRS) space, can be expressed in the form

$$\psi_{\nabla}^{\mathrm{LLRS}}(\boldsymbol{O};\{\hat{\boldsymbol{\lambda}}_q\}_{q=1}^{2},\rho) = \frac{1}{T}\left[\nabla_{\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2}^{(0,\rho)} \ln \frac{p(\boldsymbol{O}|\boldsymbol{\lambda}_1)}{p(\boldsymbol{O}|\boldsymbol{\lambda}_2)}\Big|_{\substack{\boldsymbol{\lambda}_1=\hat{\boldsymbol{\lambda}}_1 \\ \boldsymbol{\lambda}_2=\hat{\boldsymbol{\lambda}}_2}}\right] = \frac{1}{T}\begin{bmatrix} \ln p(\boldsymbol{O}|\hat{\boldsymbol{\lambda}}_1) - \ln p(\boldsymbol{O}|\hat{\boldsymbol{\lambda}}_2) \\ \nabla_{\boldsymbol{\lambda}_1} \ln p(\boldsymbol{O}|\boldsymbol{\lambda}_1)\big|_{\hat{\boldsymbol{\lambda}}_1} \\ -\nabla_{\boldsymbol{\lambda}_2} \ln p(\boldsymbol{O}|\boldsymbol{\lambda}_2)\big|_{\hat{\boldsymbol{\lambda}}_2} \\ \vdots \\ \mathrm{vec}\left(\nabla_{\boldsymbol{\lambda}_1}^{\rho} \ln p(\boldsymbol{O}|\boldsymbol{\lambda}_1)\Big|_{\hat{\boldsymbol{\lambda}}_1}\right) \\ -\mathrm{vec}\left(\nabla_{\boldsymbol{\lambda}_2}^{\rho} \ln p(\boldsymbol{O}|\boldsymbol{\lambda}_2)\Big|_{\hat{\boldsymbol{\lambda}}_2}\right) \end{bmatrix}. \tag{4.43}$$

The mapping used by Jaakkola and Haussler is a special case of (4.42) when only the first derivative is considered and is also known as Fisher mapping or Fisher score. Its expected value with respect to an observation $\boldsymbol{o}_t$ is zero, and the mapping can be written in the form

$$\psi_{\nabla}^{\mathrm{Fisher}}(\boldsymbol{O};\hat{\boldsymbol{\lambda}}) = \frac{1}{T}\nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{O}|\boldsymbol{\lambda})\Big|_{\hat{\boldsymbol{\lambda}}}\,. \tag{4.44}$$

Further, let's analyze the first-order derivatives with respect to mean and covariance of a GMM

(hence, the Fisher score). These are given by

$$\nabla_{\boldsymbol{\mu}_m} \ln p(\boldsymbol{O}|\boldsymbol{\lambda}) = \sum_{t=1}^{T} \gamma_m(t) \boldsymbol{C}_m^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_m)^{\mathrm{T}} = \eta_m (\varepsilon_m(\boldsymbol{o}) - \boldsymbol{\mu}_m)^{\mathrm{T}} \boldsymbol{C}_m^{-1} \,, \tag{4.45}$$

$$\begin{aligned} \nabla_{\boldsymbol{C}_m} \ln p(\boldsymbol{O}|\boldsymbol{\lambda}) &= \sum_{t=1}^{T} \frac{\gamma_m(t)}{2} \left[-\boldsymbol{C}_m^{-1} + \boldsymbol{C}_m^{-1} (\boldsymbol{\mu}_m - \boldsymbol{o}_t)(\boldsymbol{\mu}_m - \boldsymbol{o}_t)^{\mathrm{T}} \boldsymbol{C}_m^{-1} \right] \\ &= \frac{\eta_m}{2} \left[-\boldsymbol{C}_m^{-1} + \boldsymbol{C}_m^{-1} [\hat{\boldsymbol{C}}_m + (\boldsymbol{\mu} - \varepsilon_m(\boldsymbol{o}))(\boldsymbol{\mu} - \varepsilon_m(\boldsymbol{o}))^{\mathrm{T}}] \boldsymbol{C}_m^{-1} \right] \,, \end{aligned} \tag{4.46}$$

where $\gamma_m(t)$ is the m^{th} Gaussian's posterior specified in (3.4), $\eta_m = \sum_{t=1}^{T} \gamma_m(t)$,

$$\hat{\boldsymbol{C}}_m = \boldsymbol{\varepsilon}_m^2(\boldsymbol{o}) - \boldsymbol{\varepsilon}_m(\boldsymbol{o}) \boldsymbol{\varepsilon}_m^{\mathrm{T}}(\boldsymbol{o}), \tag{4.47}$$

and $\boldsymbol{\varepsilon}_m(\boldsymbol{o})$, $\boldsymbol{\varepsilon}_m^2(\boldsymbol{o})$ are the first and the second moment, related to the m^{th} Gaussian of a GMM, defined in (3.6). It is easy to see that the derivatives vanish when the data perfectly fit the model thus $\boldsymbol{\mu}_m \equiv \boldsymbol{\varepsilon}_m(\boldsymbol{o})$ and $\boldsymbol{C}_m \equiv \hat{\boldsymbol{C}}_m$ for $m = 1, \ldots, M$, where M is the number of GMM components. When data lie away from the model, absolute values of gradients in (4.45) and (4.46) increase, on the other hand, the closer are the data to the model, the smaller are the absolute values of gradients (under the assumption, that the amount of data does not vary significantly – consider the influence of η_m – however, this problem is solved involving the normalization term $1/T$). Hence, we are able to measure the data deviation from the model. As already mentioned, the MAP adaptation is utilized to acquire speaker dependent GMMs. When the amount of training data is small, the UBM is shifted to the speaker's direction only partially. Therefore an additional information about the direction of the data location is truly useful. The LLRS space takes into consideration not only the deviation from the speaker's GMM, but also from some other model $\hat{\boldsymbol{\lambda}}_2$ – often represented by the UBM. UBM is the same for all the speakers, therefore it can be regarded as an anchor point in the score space – a tool pointing to separate, speaker dependent data clusters.

It should be stated, that (4.46) results in a matrix. To be able to train the SVM, the matrix should be rearranged to a vector (row-wise, eventually column-wise) utilizing the function $\mathrm{vec}(\cdot)$. If only diagonal covariances are assumed, the function $\mathrm{vec}(\cdot)$ may be replaced by the function $\mathrm{diag}(\cdot)$. Note that the term η_m associated with each Gaussian can be regarded as an alternative to the normalization term T.

4.5 Derivative Kernels

4.5.1 Basic Derivative Kernel (BDK)

BDK is related to the kernel in equation (4.2). When utilizing the Fisher mapping (4.44) the normalization matrix $\boldsymbol{G}^{-1}$ from (4.3) represents the Fisher Information (FI), which plays an important role in many scientific branches. Resuming the preceding analysis of Fisher score related to generative models, the FI could be interpreted as the amount of information that a sample provides about the value of an unknown parameter $\boldsymbol{\lambda}$ [56]. However, the FI is less significant in our task as its only purpose is to ensure the correctness of the dot product in the score space. Note that when training a SVM model of *one particular speaker*, the score space vectors supplied to the training are build *only* upon MAP-adapted model of this (target) speaker and all the utterances of all the participating speakers (for LLRS space in (4.43) also UBM is involved). Loosely speaking, no other models than the target speaker model are involved in the training phase of one SVM model. Since the gradients have to be computed for each GMM and each participating speaker when training a SVM, the computational

demands may be enormous. In order to train SVMs for N GMMs with M Gaussians of dimension D and having S speakers in the development set, number of gradients to be computed is $N \times S$. Assuming the Fisher mapping the dimension of each supervector will be $M \times D$. Since it is usual to have thousands of target speakers (thus thousands of GMMs to be trained) and another thousands of speakers in the development set, the training becomes greatly time consuming.

4.5.2 Generalized Derivative Kernel (GDK)

GDK was introduced in [57]. It is based on the GLDS mapping discussed in Section 4.2 and on the Fisher score, where only derivatives with respect to the mean are considered. Hence, (4.44) can be rewritten to the form

$$\psi_{\nabla}^{\mathrm{GDK}}(\boldsymbol{O};\tilde{\boldsymbol{\lambda}}) = \frac{1}{\tilde{\eta}_i}\sum_{t=1}^{T}\tilde{\gamma}_i(t)\tilde{\boldsymbol{C}}_i^{-1}(\boldsymbol{\varphi}(\boldsymbol{o}_t)-\tilde{\boldsymbol{\mu}}_i)^{\mathrm{T}}\,, \tag{4.48}$$

where $\tilde{\boldsymbol{\lambda}} = \{\tilde{\omega}_m, \tilde{\boldsymbol{\mu}}_m, \tilde{\boldsymbol{C}}_m\}_{m=1}^{M}$ are GMM parameters related to transformed features in the new space generated by the vector function $\boldsymbol{\varphi}(\cdot)$ defined in (4.9). The normalization term changes now to $\tilde{\eta}_m = \sum_{t=1}^{T}\tilde{\gamma}_m(t)$. After substituting (4.48) into (4.2) we get

$$\begin{aligned} K(\boldsymbol{O}^i,\boldsymbol{O}^j,\tilde{\boldsymbol{\lambda}}) &= \psi_{\nabla}^{\mathrm{GDK}}(\boldsymbol{O}^i;\tilde{\boldsymbol{\lambda}})^{\mathrm{T}}\,\boldsymbol{G}\,\psi_{\nabla}^{\mathrm{GDK}}(\boldsymbol{O}^j;\tilde{\boldsymbol{\lambda}}) \\ &= \sum_{m=1}^{M}\frac{1}{\tilde{\eta}_{im}\tilde{\eta}_{jm}}\sum_{t=1}^{T_i}\sum_{s=1}^{T_j}\tilde{\gamma}_m(t)\tilde{\gamma}_m(s)k_m(\boldsymbol{o}_{it},\boldsymbol{o}_{js}) \end{aligned} \tag{4.49}$$

$$k_m(\boldsymbol{o}_i,\boldsymbol{o}_j) = (\boldsymbol{\varphi}(\boldsymbol{o}_i)-\tilde{\boldsymbol{\mu}}_m)^{\mathrm{T}}\tilde{\boldsymbol{C}}_m^{-1}\boldsymbol{G}_m\tilde{\boldsymbol{C}}_m^{-1}(\boldsymbol{\varphi}(\boldsymbol{o}_j)-\tilde{\boldsymbol{\mu}}_m)\,. \tag{4.50}$$

There are several issues concerning the solution of (4.49). For $\boldsymbol{\varphi}(\boldsymbol{o}_i) = \boldsymbol{o}_i$, we get identical kernel function to the BDK, but it is merely unfeasible to construct a GMM for other choices of $\boldsymbol{\varphi}(\cdot)$ that maps features to high-dimensional vectors (e.g. the monomial expansion as in (4.10)). Therefore, several approximations and adjustments are needed, for details see [57].

4.6 Conclusion and Remarks

It is quite difficult to establish the best or most suitable mapping/kernel from the approaches described above. Each of the techniques has its benefits and disadvantages. Attempts to study the complementarity of parametric and derivative kernels have already been carried out [41]. After meeting some assumptions, derivative mappings can be seen as gradient ascent updates of parametric ones. This is quite straightforward if considering the discussion of the Fisher mapping in Section 4.4. Hence, the complementarity of such two kernels can be anticipated. Actually, the information carried by supervectors based on GMM means and supervectors based on adaptation matrices should be uncorrelated too. Mean supervector kernels try to delimitate the part of the acoustic space specific for the speaker, whereas the derivative kernels and kernels based on adaptation matrices represent a "data error" from which the generative model suffer. Thus, a combination of such systems would be suitable.

Overwhelming majority of discussed mappings and kernels where utilized in the concept of SVM. Nevertheless, through the time several other machines were developed. E.g. the Sparse Kernel Logistic Regression (SKLR) proposed in [58], which models directly the posterior probabilities of the class membership, or the learning algorithm in [59], which has the advantage that it does not suffer from the strict Mercer's kernel restrictions defined in (2.38).

In this chapter only MLLR based kernels have been discussed, but generally, supervectors can be build on any other transformation matrix that possess a unique information about the speaker's identity – e.g. fMLLR matrix introduced in Section 3.4 [60].

Chapter 5

Normalization Techniques And Whitening

In this chapter several frequently utilized normalization techniques used in the framework of supervectors will be presented. At first methods focused on feature vector normalization will be discussed. One of the crucial problems in speaker verification systems is the choice of the verification threshold. The task is influenced by many factors making more difficult to correctly tune the threshold value. Namely: inaccurate speaker models, similar voices of speakers, inter and intra speaker variabilities, operating conditions, and many other difficulties, which reflect themselves more or less in the verification score. Subsequently, common methods for the score normalization will be reviewed, and at the end a method dealing with compensation of channel variability will be discussed.

5.1 Feature Space Whitening (FSW)

FSW deals with the fact that the dot product arising in all the kernel machines is not invariant to linear transformations. Consider a set of N two dimensional vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. Let μ_1, μ_2 and σ_1^2, σ_2^2 be the expectation and variance of the first and second dimension of the vectors in $\boldsymbol{X}$, respectively. Assuming e.g. $\mu_1 >> \mu_2$ and/or $\sigma_1^2 >> \sigma_2^2$ may lead to the domination of the first dimension in the dot product of any two vectors from $\boldsymbol{X}$ reducing the dimensionality of the space to one [61]. Hence, it is desirable to normalize elements of the vectors to zero mean and unit variance – whiten the data. The unit variance normalization is performed employing the matrix $\boldsymbol{G}$ introduced in (4.3) and the kernel function in the form of equation (4.2).

5.2 Rank Normalization (RN)

RN deals with the same problem as FSW, but in a different manner. The elements of feature vectors are adaptively rescaled to obtain approximately the uniform distribution [53]. The procedure is as follows:

- all the utterances of impostors in the background population and the given speaker utterance are mapped to a supervector,
- the elements of supervectors are sorted along each dimension,
- value of each element in the supervector of the given speaker is replaced by its rank in the sorted list (along each dimension).

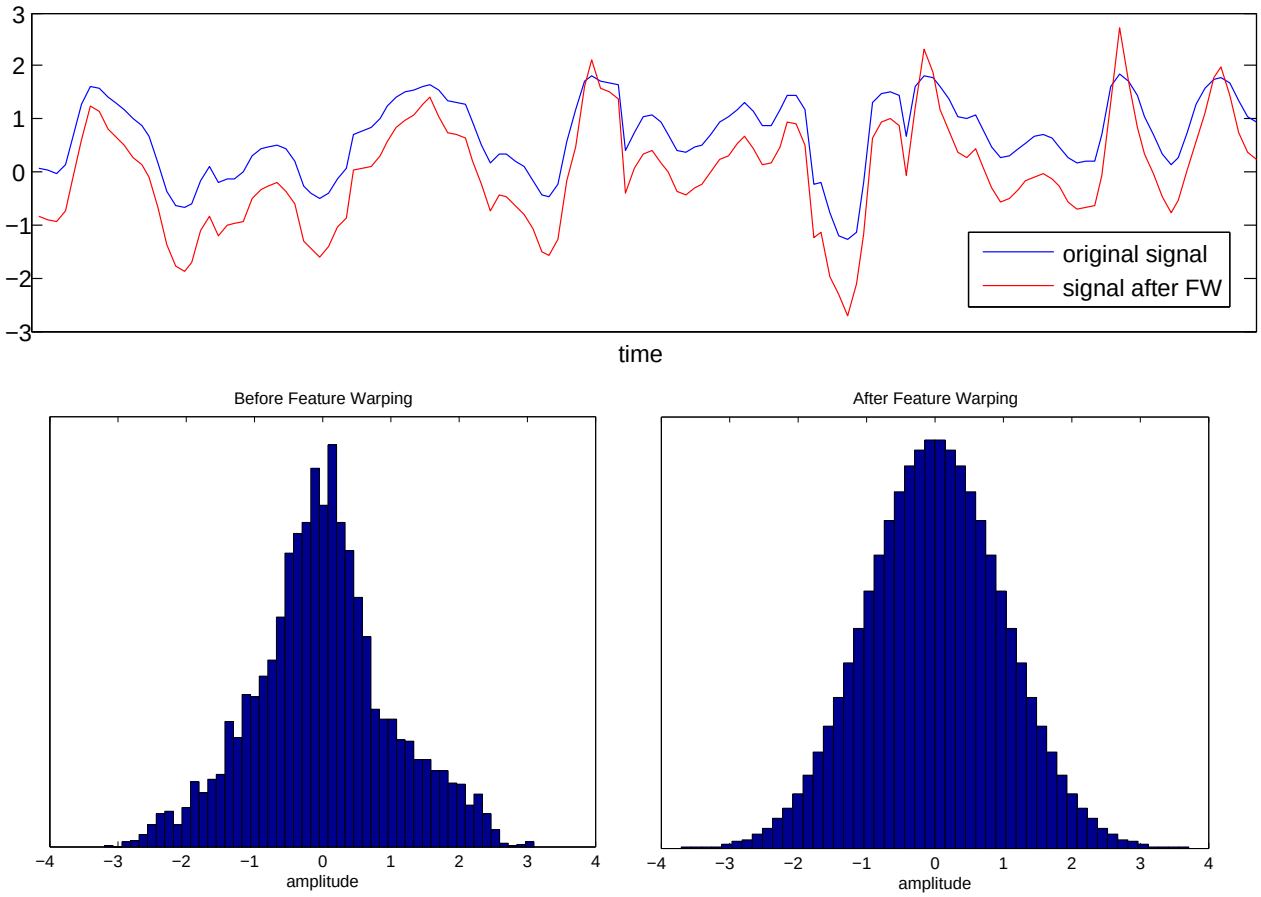

Figure 5.1: Feature Warping. A part of an one dimensional signal before and after Feature Warping is shown along with respective histograms of amplitudes of time samples. Note that the "shape" of the signal is preserved, but time samples are non-linearly scaled.

It is useful to further normalize the vectors along each dimension to a suitable interval, e.g. $[0, 1]$.

5.3 Feature Warping (FW)

FW was introduced in [62]. It performs a mapping of the distribution of a feature stream to the standard normal probability distribution. A sliding window of fixed length ν (determined empirically) is used. Firstly, all the feature vectors in the window are rank-normalized along each dimension. The feature vectors in the window are used as the background population (see discussion of RN above). Second, each entry of the feature vector is normalized by ν, and replaced by the output of the normal cumulative distribution function

$$F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-t^2/2} \mathrm{d}t. \tag{5.1}$$

A generalization of FW to any probability distribution is straightforward, one has to replace the normal cumulative distribution function by any other cumulative distribution function. The goal is to provide a more consistent distribution of feature vectors across distinct recording environments [62]. These technique is frequently used mainly in the process of feature extraction (MFCC, PLP), and proves to be of substantial importance [63] mainly when variations in channel conditions are

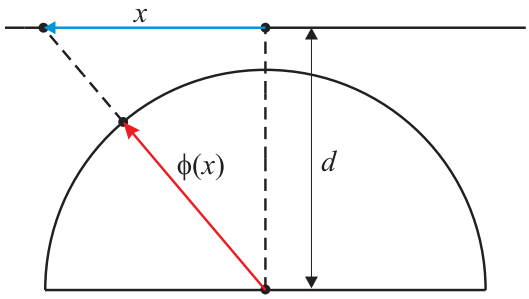

Figure 5.2: Mapping of an one dimensional feature x onto the surface of a unit hypersphere. Note that the mapped feature $\phi(x)$ is of one dimension higher than the original x.

significant. An example of FW of a time sequence of samples is depicted in Figure 5.1, more examples on FW of feature vectors along each dimension can be found in Appendix F.

5.4 Spherical Normalization (SN)

SN arises from projections used by cartographers. It can be thought of as a transformation that maps each feature vector onto the surface of a unit hypersphere embedded in a space one dimension higher then the feature vector itself [61]. The value of the kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can take small as well as large values depending on $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, and thus make the Hessian (2.36) badly conditioned or, in the worst case, even singular – especially for polynomial kernels with high powers [64]. Such a situation can occur even if elements of each feature vector were already normalized to a narrow interval (e.g. $[-1, 1]$), because of the high dimensionality of supervectors. Therefore, the SN is applied in advantage before the evaluation of the kernel function. The form of SN mapping $\phi : R^n \mapsto R^{n+1}$ used in [61] has the form

$$\phi(\boldsymbol{x}) = \frac{1}{\sqrt{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} + d^2}} \begin{bmatrix} \boldsymbol{x} \\ d \end{bmatrix}, \tag{5.2}$$

where d is an empirically set constant (for whitened data a reasonable choice can be $d = 1$). One could also utilize the L_2 vector norm $\|\boldsymbol{x}\|_2$ instead of SN, hence $\phi(\boldsymbol{x}) = \boldsymbol{x}/\|\boldsymbol{x}\|$ but this would lead to information loss (two distinct vectors $\boldsymbol{x}$ and $2\boldsymbol{x}$ would become one). A 1D example of SN is depicted in Figure 5.2.

5.5 Within Class Covariance Normalization (WCCN)

Within Class Covariance Normalization [65] matrix is given as

$$\boldsymbol{W} = \frac{1}{S}\sum_{s=1}^{S}\frac{1}{N_s}\sum_{i=1}^{N_s}(\boldsymbol{x}_{si} - \bar{\boldsymbol{x}}_s)(\boldsymbol{x}_{si} - \bar{\boldsymbol{x}}_s)^{\mathrm{T}}, \;\; \bar{\boldsymbol{x}}_s = \frac{1}{N_s}\sum_{i=1}^{N_s}\boldsymbol{x}_{si}, \tag{5.3}$$

where $\boldsymbol{\Lambda}_s = \{\boldsymbol{x}_{si}\}_{i=1}^{N_s}$ is the set of N_s vectors belonging to the speaker s, and S is the number of speakers. The idea behind WCCN is to weight the dimensions of the feature space according to the inverse of the WCCN matrix, and thus to decrease the influence of the directions with high intra-speaker variability when evaluating the linear function $\boldsymbol{x}_i\boldsymbol{W}^{-1}\boldsymbol{x}_j$.

5.6 Zero and Test Normalizations

Zero and Test Normalizations are related to the verification score rather than to feature vectors. The Zero Normalization (Z-norm) method was proposed in [26] and its purpose is to ensure zero mean and unit deviation of scores for speech of impostor speakers. The Z-normalized score for s^{th} speaker (target model) and i^{th} sequence of feature vectors $\boldsymbol{O}^i$ is computed according to

$$L_{\text{Z}}(\boldsymbol{O}^i|s) = \frac{L(\boldsymbol{O}^i|s) - \mu_{\text{I}}}{\sigma_{\text{I}}}\,, \tag{5.4}$$

where $L(\boldsymbol{O}^i|s)$ denotes the score for speaker s and utterance i, μ_{I} and σ_{I} are estimated from the background population of impostors. The speaker model is scored against impostor utterances yielding a set of scores related to the given speaker model. This set is used to compute the Z-norm parameters μ_{I} and σ_{I}. Note that the parameters for Z-norm can be estimated off-line during the training of a speaker model.

The Test Normalization (T-norm) introduced in [66] is computed using the same formula as Z-norm. The difference consists in estimation of μ_{I} and σ_{I}. Now, a set of impostor models is chosen beforehand. The T-norm parameters are then estimated from the set of scores obtained during the verification phase, where all the impostor models are scored against the i^{th} utterance. As expected, the disadvantage of T-norm are the increased computational demands. Note that the T-norm is very similar to the cohort normalization [67], but the score is now in addition normalized by the standard deviation. For instructions how to adequately choose the cohort speakers for T-norm see e.g. [68].

These two techniques can be combined (as done very often) into the so-called ZT-norm, where Z-norm is followed by the T-norm yielding a superior performance [69].

With time ZT-norm became a separate module used regularly in speaker verification systems regardless on other normalization techniques as the last tool to adjust the score. However, the choice of the impostor/cohort speakers is still an alchemy even if some techniques were already proposed, e.g. [68].

5.7 Nuisance Attribute Projection (NAP)

NAP was suited for the concept of SVMs and supervectors [70, 71, 42]. It reduces the influence of the channel variability projecting out the supervector dimensions that are mostly vulnerable to changes of operating conditions.

We are given a set of labeled vectors. Each group, containing vectors sharing the same label, consists of several supervectors (representations) of one individual obtained from distinct operating conditions. The task is to locate directions in which the variations between distinct representations in each of the groups are highest. These directions are then removed (projected out). It can be anticipated that the problem can be solved by an eigenvector decomposition of some covariance matrix. Now it will be shown how does the objective function and the covariance matrix look like.

The objective function to be minimized has the form [70]

$$J_{\text{NAP}}(\boldsymbol{P}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} W_{ij} \|\boldsymbol{P}(\boldsymbol{x}_i - \boldsymbol{x}_j)\|^2 \tag{5.5}$$

where N is the number of input vectors, $\boldsymbol{W} = [W_{ij}]$ is a $N \times N$ symmetric matrix of weighting factors (they can be seen as a set of labels discussed above), $\boldsymbol{P}$ is a $D \times D$ projection matrix of low rank D_p, and $D = \dim(\boldsymbol{x}_i)$. Note that $D - D_p$ is known as the *corank* of the matrix $\boldsymbol{P}$, and we will

adhere to this term also in the experiments in Section 7.6. The projection matrix $\boldsymbol{P}$ will be assumed in the form

$$\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{F}\boldsymbol{F}^{\mathrm{T}}, \tag{5.6}$$

where columns of the $D \times D_p$ matrix $\boldsymbol{F}$ span the subspace which we are going to project out, and we will put a slight restriction on the basis of the subspace in the form $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} = \boldsymbol{I}$. Thus, columns of $\boldsymbol{F}$ are orthonormal, otherwise the projection matrix would have the form $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{F}(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F})^{-1}\boldsymbol{F}^{\mathrm{T}}$ (note that the assumption that $\boldsymbol{F}$ has full column rank D_p has to be met, otherwise the inversion $(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F})^{-1}$ has to be replaced by a generalized inversion [72]). However, the objective (5.12) does not depend on the choice of the base of the subspace, it depends only on the subspace generated by this base. Hence choosing the projection $\boldsymbol{P}$ in the form (5.6) does not violate the generality of the solution of (5.5). The proof is given in Appendix A.

One of the simplest weightings presented in [70] are

- $\boldsymbol{W}_{\text{channel}}$ – $W_{ij} = 1$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ differ in the channel, 0 otherwise,
- $\boldsymbol{W}_{\text{session}}$ – $W_{ij} = 1$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ correspond to the same speaker, 0 otherwise.

The first case $\boldsymbol{W}_{\text{channel}}$ can be interpreted as the minimization of cross-channel distances, and the latter case $\boldsymbol{W}_{\text{session}}$ can be seen as the minimization of the session variability. Let $\boldsymbol{X}_s = [\boldsymbol{x}_{s1}, \ldots, \boldsymbol{x}_{sN_s}]$ be the data matrix with N_s vectors ordered in columns containing only those vectors for which $W_{ij} = 1$, and let $\boldsymbol{X} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_S]$ be the overall data matrix containing disjoint sets of all the input vectors. Hence, weights W_{ij} can be seen also as data labels, which determine the pertinence of a vector to one of the sets $\boldsymbol{X}_s$. After such an arrangement of vectors in $\boldsymbol{X}$ the weighting matrix $\boldsymbol{W}$ will have the form

$$\boldsymbol{W} = \begin{bmatrix} \mathbb{I}_{N_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbb{I}_{N_S} \end{bmatrix}, \tag{5.7}$$

where $\mathbb{I}_{N_s}$ is a $N_s \times N_s$ matrix with 1 in each entry, S is the number of unique labels. Equation (5.5) can be now rewritten as

$$\begin{aligned} \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} W_{ij}\|\boldsymbol{P}(\boldsymbol{x}_i - \boldsymbol{x}_j)\|^2 &= \sum_{s=1}^{S}\sum_{i=1}^{N_s-1}\sum_{j=i+1}^{N_s} \boldsymbol{e}_{ij}^{\mathrm{T}}\boldsymbol{X}_s^{\mathrm{T}}\boldsymbol{P}\boldsymbol{X}_s\boldsymbol{e}_{ij} = \sum_{s=1}^{S}\operatorname{tr}\left(\boldsymbol{P}\boldsymbol{X}_s\left(\sum_{i,j}\boldsymbol{e}_{ij}\boldsymbol{e}_{ij}^{\mathrm{T}}\right)\boldsymbol{X}_s^{\mathrm{T}}\right) \\ &= \operatorname{tr}\left(\boldsymbol{P}\sum_{s=1}^{S} N_s\boldsymbol{X}_s\boldsymbol{J}_s\boldsymbol{X}_s^{\mathrm{T}}\right) = \operatorname{tr}\left(\boldsymbol{P}\boldsymbol{X}\boldsymbol{J}\boldsymbol{X}^{\mathrm{T}}\right), \end{aligned} \tag{5.8}$$

where the property $\boldsymbol{P}^2 = \boldsymbol{P}$ of the projection matrix (5.6) was used ($\boldsymbol{P}$ is idempotent, projection of a projection of a vector does not change the vector anymore since the vector is already in the subspace generated by $\boldsymbol{P}$, additionally $\boldsymbol{P} = \boldsymbol{P}^{\mathrm{T}}$), $\boldsymbol{e}_{ij}$ is a zero vector with +1 in its i^{th} entry and -1 in its j^{th} entry ($\boldsymbol{X}_s\boldsymbol{e}_{ij} = \boldsymbol{x}_{si} - \boldsymbol{x}_{sj}$), and

$$\boldsymbol{J}_s = \frac{1}{N_s}\sum_{i,j}\boldsymbol{e}_{ij}\boldsymbol{e}_{ij}^{\mathrm{T}} = \boldsymbol{I}_{N_s} - \frac{1}{N_s}\boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}, \tag{5.9}$$

$\boldsymbol{I}_{N_s}$ is $N_s \times N_s$ identity matrix, and

$$\boldsymbol{J} = \operatorname{DIAG}(\boldsymbol{W}\boldsymbol{1}) - \boldsymbol{W} = \begin{bmatrix} N_1\boldsymbol{J}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & N_S\boldsymbol{J}_S \end{bmatrix}, \tag{5.10}$$

where the function DIAG($\boldsymbol{z}$) takes an input vector $\boldsymbol{z}$ and constructs a diagonal matrix $\boldsymbol{Z}$ with $\boldsymbol{z}$ on its diagonal. Realizing that

$$\boldsymbol{X}_s \boldsymbol{J}_s \boldsymbol{X}_s^{\mathrm{T}} = \sum_{n=1}^{N_s} (\boldsymbol{x}_{sn} - \bar{\boldsymbol{x}}_s)(\boldsymbol{x}_{sn} - \bar{\boldsymbol{x}}_s)^{\mathrm{T}}, \text{ with } \bar{\boldsymbol{x}}_s = \sum_{n=1}^{N_s} \boldsymbol{x}_{sn}, \tag{5.11}$$

we can state that the matrix (5.10) does the centering of vectors in each set $\boldsymbol{X}_s$. Thus, the matrix $\boldsymbol{C}_{\mathrm{W}} = \boldsymbol{X}\boldsymbol{J}\boldsymbol{X}^{\mathrm{T}}$ is in fact the weighted sum of within covariances of each labeled set $\boldsymbol{X}_s$ weighted according to the number of vectors it contains. The objective function (5.5) takes now the form

$$J_{\mathrm{NAP}}(\boldsymbol{P}) = \mathrm{tr}(\boldsymbol{P}\boldsymbol{C}_{\mathrm{W}}). \tag{5.12}$$

Substituting (5.6) into (5.12) and keeping only the part dependent on $\boldsymbol{F}$ we get a new objective function

$$J_{\mathrm{NAP}}(\boldsymbol{F}) = \mathrm{tr}(\boldsymbol{F}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{C}_{\mathrm{W}}) = \mathrm{tr}(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{J}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{F}) = \mathrm{tr}(\boldsymbol{Y}\boldsymbol{J}\boldsymbol{Y}^{\mathrm{T}}) = \mathrm{tr}(\tilde{\boldsymbol{C}}_{\mathrm{W}}^{F}), \tag{5.13}$$

where $\boldsymbol{Y} = \boldsymbol{F}^{\mathrm{T}}\boldsymbol{X}$ are the coordinates of vectors from the set $\boldsymbol{X}$ in the subspace formed by columns of $\boldsymbol{F}$. It is now obvious that minimizing (5.5) is equivalent to maximizing the trace of the within covariance matrix in the subspace that we wish to project out.

Using the Lagrange function with a symmetric matrix $\boldsymbol{\Lambda}$ (since the restriction is symmetric) [73] of Lagrange multipliers we get

$$L_F = \mathrm{tr}(\ (\boldsymbol{I} - \boldsymbol{F}\boldsymbol{F}^{\mathrm{T}})\boldsymbol{C}_{\mathrm{W}}\ ) + \mathrm{tr}(\ \boldsymbol{\Lambda}(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} - \boldsymbol{I})\ ) \tag{5.14}$$

Computing the derivative with respect to $\boldsymbol{F}$ and setting it to zero we get

$$\begin{aligned} -2\boldsymbol{C}_{\mathrm{W}}\boldsymbol{F} + 2\boldsymbol{F}\boldsymbol{\Lambda} &= 0 \\ \boldsymbol{C}_{\mathrm{W}}\boldsymbol{F} &= \boldsymbol{F}\boldsymbol{\Lambda}, \end{aligned}$$

Since $\boldsymbol{C}_{\mathrm{W}}$ is a symmetric positive semi-definite matrix its eigenvalues are nonnegative and its eigenvectors are orthonormal. Thus, the restriction $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} = \boldsymbol{I}$ is satisfied when columns of $\boldsymbol{F}$ are eigenvectors of $\boldsymbol{C}_{\mathrm{W}}$, and (5.12) is minimized when these eigenvectors correspond to D_p largest eigenvalues (highest variance is projected out). In fact, the concept of NAP is equivalent to the concept of Principal Component Analysis (PCA) [74] used mainly for dimensionality reduction, where an eigenvalue decomposition of some covariance matrix is carried out, and the eigenvectors corresponding to largest eigenvalues are used to form the columns of the transformation matrix.

The problem with $\boldsymbol{W}_{\mathrm{channel}}$ is that it tries to put together also different speakers who can be considered as distinct representations of one speaker/voice differing in the channel. This can be solved by centralizing the supervectors of one speaker in advance, and thereby normalizing out the identity of a speaker (after some assumptions are met). However, following the previous discussion, such an approach is equivalent to the one involving the matrix $\boldsymbol{W}_{\mathrm{session}}$, where (as was shown above) the problem of centralization is solved implicitly.

Note that the dimension of the feature vector $\boldsymbol{x}$ after being projected ($\hat{\boldsymbol{x}} = \boldsymbol{P}\boldsymbol{x}$) is preserved – the vector $\boldsymbol{x}$ is projected on some subspace in the high dimensional space. The formulation of NAP was originally proposed for SVMs, but some attempts have been made to extended it also to conventional GMMs [75].

Chapter 6

Factor Analysis Based Techniques

In last years Factor Analysis (FA) based techniques gained on popularity in the task of speaker recognition. Progressive methods as Joint Factor Analysis (JFA) [76], closely related concept of i-vectors [5] or Probabilistic Linear Discriminant Analysis (PLDA) [8] are all based on FA. FA was integrated into the task of speaker recognition when supervectors (mostly based on GMM parameters) were introduced. Since supervectors are of substantially high dimension methods to reduce their dimension and/or bind the parameters in a supervecetor were requested. First techniques dealing with the dimensionality reduction of supervectors in the sense of subspace estimation were Nuisance Attribute Projection (NAP) described in Section 5.7 and JFA [77]. While NAP was suited for support vector machines and addressed the channel compensation, JFA was focusing also on the within-speaker variability. Both within- and between-speaker subspaces were estimated *jointly* at the same time. Lately, in [78] it was shown how to decouple the estimation of both subspaces. At first the between-speaker subspace is estimated, and subsequently the within-speaker covariance is decomposed and the within-speaker subspace is determined. However, since the channel component of a supervector does still contain a speaker information [5] JFA was modified to the concept of i-vectors. All the methods are going to be reviewed in following sections.

6.1 Factor Analysis (FA)

Let $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ be a set of N feature vectors (observations) of dimension D_x and let $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^N$ be a set of their latent representations of dimension D_y (often $D_y \ll D_x$). Then, *Factor Analysis* is a latent linear Gaussian model of the form

$$\boldsymbol{x}_i = \boldsymbol{B}\boldsymbol{y}_i + \boldsymbol{\epsilon}_i, \tag{6.1}$$

where $\boldsymbol{B}$ is a $D_x \times D_y$ transformation matrix, and an assumption has been made that vectors $\boldsymbol{x}_i$ were normalized in advance to have zero mean, otherwise the formula would contain also a mean term $\boldsymbol{\mu}$, e.g. $\boldsymbol{x}_i = \boldsymbol{\mu} + \boldsymbol{B}\boldsymbol{y}_i + \boldsymbol{\epsilon}_i$. The term $\boldsymbol{\epsilon}_i$ represents some residual noise contained in the data. In FA an assumption is met that the variation in feature vectors can be explained in a sufficient amount by variations of low dimensional hidden variables $\boldsymbol{y}_i$.

In order to enable and facilitate the estimation of unknown parameters restrictions on distributions

of $\boldsymbol{y}_i$ and $\boldsymbol{\epsilon}_i$ are laid. We will assume that

$$\boldsymbol{y}_i, \boldsymbol{\epsilon}_i \text{ are independent and identically distributed (iid)}, \tag{6.2}$$
$$\boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}), \tag{6.3}$$
$$\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} + \boldsymbol{\Sigma}), \text{ alternatively } \boldsymbol{x}_i|\boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{B}\boldsymbol{y}_i, \boldsymbol{\Sigma}), \tag{6.4}$$
$$\boldsymbol{y}_i|\boldsymbol{x}_i \sim \mathcal{N}(E_y[\boldsymbol{y}_i|\boldsymbol{x}_i], \boldsymbol{C}_{yy|x}), \tag{6.5}$$

where $E_y[\cdot]$ denotes the expectation over the latent variable $\boldsymbol{y}$. In order to get parameters of the distribution specified in (6.5) one can utilize the Bayes rule, where equations (6.3) and (6.4) are plugged in, what gives

$$\boldsymbol{C}_{yy|x} = \boldsymbol{I} - \boldsymbol{B}^{\mathrm{T}}(\boldsymbol{\Sigma} + \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}})^{-1}\boldsymbol{B} = (\boldsymbol{B}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{B} + \boldsymbol{I})^{-1}. \tag{6.6}$$

$$E_y[\boldsymbol{y}_i|\boldsymbol{x}_i] = \boldsymbol{C}_{yy|x}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_i, \tag{6.7}$$

where the inverse identity (C.1) was used in (6.6). We see that the covariance of an estimate of the latent variable $\boldsymbol{y}_i$ given an observation $\boldsymbol{x}_i$ decreases when the noise corruption gets smaller than the covariance handled by $\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}$ (compare to $1 - \frac{\sigma_y^2}{\sigma_\epsilon^2 + \sigma_y^2}$).

6.1.1 Training

Since the set of latent variables $\boldsymbol{Y}$ as well as parameters $\Theta = \{\boldsymbol{B}, \boldsymbol{\Sigma}\}$ are unknown Expectation Maximization (EM) algorithm has to be involved in order to train the model. The objective function in EM to be optimized is given as

$$Q(\Theta, \Theta_{\text{old}}) = E_y\left[\ln p(\boldsymbol{X}, \boldsymbol{Y}|\Theta)|\boldsymbol{X}, \Theta_{\text{old}}\right] = E_y\left[\sum_{i=1}^{N}\left(\ln p(\boldsymbol{x}_i|\boldsymbol{y}_i, \Theta) + \ln p(\boldsymbol{y}_i)\right)|\boldsymbol{X}, \Theta_{\text{old}}\right], \tag{6.8}$$

Both distributions in (6.8) are known (defined in (6.3) and (6.4)), thus

$$Q(\Theta, \Theta_{\text{old}}) = \text{const} - \frac{1}{2}\sum_{i=1}^{N} E_y\left[\ln|\boldsymbol{\Sigma}| + (\boldsymbol{x}_i - \boldsymbol{B}\boldsymbol{y}_i)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \boldsymbol{B}\boldsymbol{y}_i)|\boldsymbol{X}, \Theta_{\text{old}}\right] \tag{6.9}$$

where everything not dependent on Θ is contained in the const part. In order to maximize (6.9) we use identities [79]

$$\frac{\partial}{\partial \boldsymbol{B}}(\boldsymbol{x} - \boldsymbol{B}\boldsymbol{y})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{B}\boldsymbol{y}) = -2\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{B}\boldsymbol{y})\boldsymbol{y}^{\mathrm{T}}, \tag{6.10}$$

$$\frac{\partial}{\partial \boldsymbol{\Sigma}}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{y} = -\boldsymbol{\Sigma}^{-\mathrm{T}}\boldsymbol{x}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{\Sigma}^{-\mathrm{T}}, \tag{6.11}$$

$$\frac{\partial}{\partial \boldsymbol{\Sigma}}\ln|\boldsymbol{\Sigma}| = \boldsymbol{\Sigma}^{-\mathrm{T}}, \tag{6.12}$$

and according to the Leibniz integral rule we get

$$\frac{d}{dx}\int_{y_0}^{y_1} f(x,y)dy = \int_{y_0}^{y_1}\frac{\partial}{\partial x}f(x,y)dy, \text{ thus } \frac{d}{d\boldsymbol{x}}E_y[f(\boldsymbol{x},\boldsymbol{y})] = E_y\left[\frac{\partial}{\partial \boldsymbol{x}}f(\boldsymbol{x},\boldsymbol{y})\right]. \tag{6.13}$$

Finally, before the the very beginning of deriving the update formulas note that

$$E_y[\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i] = \boldsymbol{C}_{yy|x} + E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]E_y[\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]. \tag{6.14}$$

Taking derivatives of objective function (6.9) we obtain

$$\frac{\partial Q(\Theta,\Theta_{\text{old}})}{\partial \boldsymbol{B}} = \sum_{i=1}^{N} E_y\left[\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_i\boldsymbol{y}_i^{\mathrm{T}} - \boldsymbol{\Sigma}^{-1}\boldsymbol{B}\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{X},\Theta_{\text{old}}\right] = \\ = \sum_{i=1}^{N}\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_i E_y[\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i] - \boldsymbol{\Sigma}^{-1}\boldsymbol{B}E_y[\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]\right) \tag{6.15}$$

$$\frac{\partial Q(\Theta,\Theta_{\text{old}})}{\partial \boldsymbol{\Sigma}} = -\frac{1}{2}\sum_{i=1}^{N} E_y\left[\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \boldsymbol{B}\boldsymbol{y}_i)(\boldsymbol{x}_i - \boldsymbol{B}\boldsymbol{y}_i)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}|\boldsymbol{X},\Theta_{\text{old}}\right] = \\ = -\frac{1}{2}\sum_{i=1}^{N}\left(\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\boldsymbol{B}E_y[\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]\boldsymbol{B}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1} + \right. \\ \left. + \boldsymbol{\Sigma}^{-1}\boldsymbol{x}_i E_y[\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]\boldsymbol{B}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-1}\boldsymbol{B}E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\right), \tag{6.16}$$

where for clarity the conditional part in the expectation $E_y[\boldsymbol{y}_i|\boldsymbol{x}_i,\Theta_{\text{old}}]$ was shortened. Setting them to zero we get following update formulas

$$\boldsymbol{B} = \left(\sum_{i=1}^{N}\boldsymbol{x}_i E_y[\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]\right)\left(\sum_{i=1}^{N} E_y[\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]\right)^{-1}, \tag{6.17}$$

$$\boldsymbol{\Sigma} = \boldsymbol{B}\boldsymbol{C}_{yy|x}\boldsymbol{B}^{\mathrm{T}} + \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)^{\mathrm{T}},\ \hat{\boldsymbol{x}}_i = \boldsymbol{B}E_y[\boldsymbol{y}_i|\boldsymbol{x}_i], \tag{6.18}$$

where $\hat{\boldsymbol{x}}_i = \boldsymbol{B}E_y[\boldsymbol{y}_i|\boldsymbol{x}_i] \approx \boldsymbol{B}\boldsymbol{y}_i$ is the reconstructed vector $\boldsymbol{x}_i$, and the term $\boldsymbol{B}\boldsymbol{C}_{yy|x}\boldsymbol{B}^{\mathrm{T}}$ represents the latent covariance expressed in the feature space. We can conclude that $\boldsymbol{\Sigma}$ explains the residual variance and in addition it grows also in regions where the covariance of latent variables given observed $\boldsymbol{x}_i$ is high (regions of high degree of uncertainty of observed- to latent-variable mapping). Realizing that the symmetry

$$\textstyle\sum_{i=1}^{N}\boldsymbol{B}E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]\boldsymbol{x}_i^{\mathrm{T}} = \sum_{i=1}^{N}\boldsymbol{x}_i E_y[\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]\boldsymbol{B}^{\mathrm{T}} = \sum_{i=1}^{N}\boldsymbol{B}E_y[\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]\boldsymbol{B}^{\mathrm{T}} =$$

$$= \left(\textstyle\sum_{i=1}^{N}\boldsymbol{x}_i E_y[\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]\right)\left(\sum_{i=1}^{N} E_y[\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]\right)^{-1}\left(\sum_{i=1}^{N} E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]\boldsymbol{x}_i^{\mathrm{T}}\right)$$

holds when we substitute (6.17) for $\boldsymbol{B}$ and using the fact that $E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]E_y[\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i] = E_y[\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i] - \boldsymbol{C}_{yy|x}$, the update formula (6.18) can be rewritten into a more efficient form

$$\boldsymbol{\Sigma} = \boldsymbol{B}\boldsymbol{C}_{yy|x}\boldsymbol{B}^{\mathrm{T}} + \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}} - \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{B}E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]\boldsymbol{x}_i^{\mathrm{T}} - \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i E_y[\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]\boldsymbol{B}^{\mathrm{T}} \\ + \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{B}(E_y[\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i] - \boldsymbol{C}_{yy|x})\boldsymbol{B}^{\mathrm{T}} = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}} - \boldsymbol{B}E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]\boldsymbol{x}_i^{\mathrm{T}}\right). \tag{6.19}$$

To summarize the training procedure given in Alg.1, the *expectation step* requires to compute (6.6), (6.7) and (6.14), which are then used in the *maximization step* to estimate new model matrices $\boldsymbol{B}$ and $\boldsymbol{\Sigma}$ according to (6.17) and (6.19). These two steps are iterated until the change in the objective function (6.8) between two successive steps is small enough or predetermined number of iterations N_{iter} was reached. To initialize the algorithm the matrices may be chosen randomly.

Note that in order to use the update formula (6.19) new estimate of $\boldsymbol{B}$ given in (6.17) must be evaluated prior to $\boldsymbol{\Sigma}$, whereas $E_y[\boldsymbol{y}_i|\boldsymbol{x}_i], i = 1,\ldots,N$ have to be the same as in (6.17) (thus computed utilizing the estimate of $\boldsymbol{B}$ from the previous step), otherwise the semi-definiteness and symmetry of the noise covariance matrix $\boldsymbol{\Sigma}$ would not be guaranteed.

Algorithm 1 FA estimation algorithm

Require: a set of centralized feature vectors $\{\boldsymbol{x}_i\}_{i=1}^N$ of dimension D_x;
dimension $D_y < D_x$ of latent representations $\boldsymbol{y}_i$;
initialization (random) matrices $\boldsymbol{B}_{\text{init}}$, $\boldsymbol{\Sigma}_{\text{init}}$;
number of iterations N_{iter}

$\boldsymbol{B} = \boldsymbol{B}_{\text{init}}$, $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\text{init}}$
$\boldsymbol{Z}_{xy} = \boldsymbol{Z}_{yy} = \boldsymbol{0}$
$\boldsymbol{C}_X = \frac{1}{N}\sum_{i=1}^N \boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}$
for $iter = 1$ to N_{iter} **do**
  // Expectation step:
  $\boldsymbol{C}_{yy|x} = (\boldsymbol{B}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{B} + \boldsymbol{I})^{-1}$
  for $i = 1$ to N **do**
    $E_y[\boldsymbol{y}_i|\boldsymbol{x}_i] = \boldsymbol{C}_{yy|x}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_i$
    $E_y[\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i] = \boldsymbol{C}_{yy|x} + E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]E_y[\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]$
    $\boldsymbol{Z}_{xy} = \boldsymbol{Z}_{xy} + \boldsymbol{x}_i E_y[\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]$
    $\boldsymbol{Z}_{yy} = \boldsymbol{Z}_{yy} + E_y[\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}|\boldsymbol{x}_i]$
  end for
  // Maximization step:
  $\boldsymbol{B} = \boldsymbol{Z}_{xy}\boldsymbol{Z}_{yy}^{-1}$
  $\boldsymbol{\Sigma} = \boldsymbol{C}_X - \frac{1}{N}\boldsymbol{B}\boldsymbol{Z}_{xy}^{\mathrm{T}}$
end for
return $\boldsymbol{B}$ and $\boldsymbol{\Sigma}$

6.1.2 Notes on FA

Until now none restrictions have been put on the form of the noise covariance matrix $\boldsymbol{\Sigma}$. The standard FA model assumes diagonal $\boldsymbol{\Sigma}$, thus

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{i=1}^N \text{diagz}\left(\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}} - \boldsymbol{B}E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]\boldsymbol{x}_i^{\mathrm{T}}\right), \tag{6.20}$$

where the function diagz() zeros all the non-diagonal elements of a matrix. The matrix $\boldsymbol{B}$ is often called the *factor loading matrix*, entries of $\boldsymbol{y}_i$ are called *factors*, and the diagonal elements of $\boldsymbol{\Sigma}$ are called *uniqueness*.

An interesting fact to note is that (6.17) minimizes also the objective function

$$J_B = \frac{1}{2}\sum_{i=1}^N \|\boldsymbol{x}_i - \boldsymbol{B}E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]\|^2 + \frac{N}{2}\text{tr}(\boldsymbol{B}\boldsymbol{C}_{yy|x}\boldsymbol{B}^{\mathrm{T}}). \tag{6.21}$$

It is straightforward to show that setting the derivative of (6.21) according to $\boldsymbol{B}$ to zero having both $E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]$ and $\boldsymbol{C}_{yy|x}$ fixed (independent of the new estimate of $\boldsymbol{B}$) and expressing $\boldsymbol{B}$ yields (6.17). It is the problem of regularized least squares and it brings new insight into the concept of FA. The iterative estimation of $\boldsymbol{B}$ consists of two steps. At first the current $\boldsymbol{B}$ is used to get the mean and the covariance of latent variables $\boldsymbol{y}_i|\boldsymbol{x}_i$ given in (6.5), and subsequently new $\boldsymbol{B}$ that minimizes (6.21) is found. Also note that the noise covariance $\boldsymbol{\Sigma}$ appears only when evaluating $E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]$ and $\boldsymbol{C}_{yy|x}$, thus it alters only the latent variables. The second term in (6.21) – the regularization term – is used to push the directions of $\boldsymbol{B}$, in which the covariance of latent variables is high, toward zero; note that

$$\text{tr}(\boldsymbol{B}\boldsymbol{C}_{yy|x}\boldsymbol{B}^{\mathrm{T}}) \approx \text{tr}\left(\frac{1}{N}\sum_i (\boldsymbol{y}_i - E_y[\boldsymbol{y}_i|\boldsymbol{x}])^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}(\boldsymbol{y}_i - E_y[\boldsymbol{y}_i|\boldsymbol{x}])\right). \tag{6.22}$$

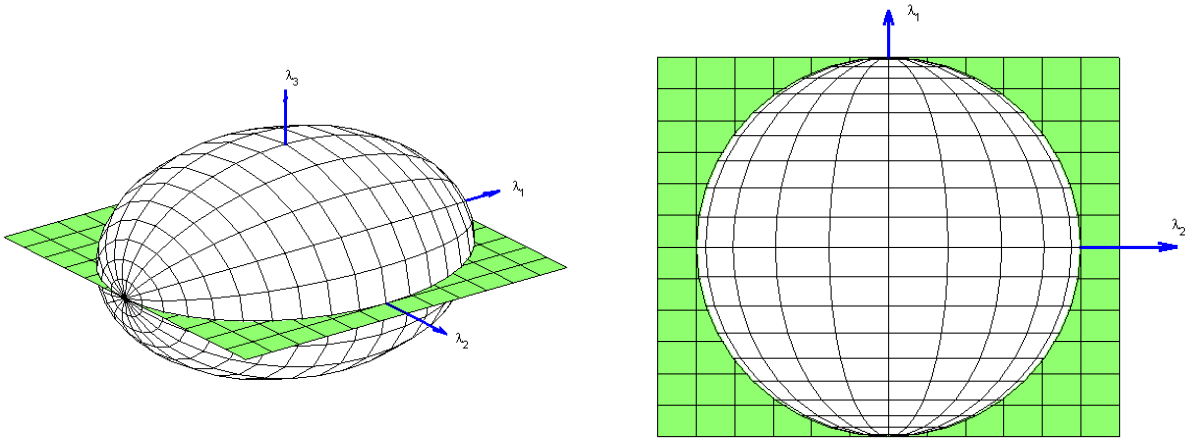

Figure 6.1: The white ellipsoid represents the spread of the sample covariance $\boldsymbol{S}$ of some three-dimensional feature vectors (i.e. the surface is given by $(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{S}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = k$, where $\boldsymbol{\mu}$ is the mean of $\boldsymbol{x}$ and $k > 0$ is some constant), and the green plane is the two-dimensional subspace formed by columns of $\boldsymbol{B}$. Eigenvalues of the sample covariance matrix $\boldsymbol{S}$ are $\lambda_1, \lambda_2, \lambda_3$, where $\lambda_1 > \lambda_2 > \lambda_3$. Note that columns of $\boldsymbol{B}$ are the leading eigenvectors of the sample covariance matrix $\boldsymbol{S}$ - this is the case in the problem of PPCA. In the second image the covariance of projected feature vectors is shown. The latent covariance $\boldsymbol{C}_{yy|x}$ is inversely proportional to λ_1, λ_2, thus the wider is the spread of feature vectors in a direction of the feature space the lower is the covariance of the latent estimate $\boldsymbol{y}_i$ in this direction.

Thus, the rank of the matrix $\boldsymbol{B}$ can decrease below D_y (assuming that the rank of the sample covariance matrix is at least D_y).

Whenever Gaussianity is involved ambiguities in rotation of coordinates occur. Since distribution of latent variables $\boldsymbol{y}_i$ is Gaussian with covariance equal to the identity matrix, the covariance of latent variables is invariant under any arbitrary rotation matrix. Loosely speaking, any rotation in the latent space does not change the solution, thus $\boldsymbol{x}_i = \boldsymbol{B}\boldsymbol{y}_i + \boldsymbol{\epsilon}_i$ and $\boldsymbol{x}_i = \hat{\boldsymbol{B}}\boldsymbol{R}\boldsymbol{y}_i + \hat{\boldsymbol{\epsilon}}_i$, where $\boldsymbol{R}$ is any rotation matrix such as $\boldsymbol{R}^{\mathrm{T}} = \boldsymbol{R}^{-1}$, results in $\boldsymbol{B} = \hat{\boldsymbol{B}}$ and $\boldsymbol{\Sigma} = \hat{\boldsymbol{\Sigma}}$. This brings a lot of difficulties when attempting to assign a meaning to the latent variables (factors).

In [80] it was shown that if $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$ is isotropic, the solution (in the sense of maximum likelihood of $p(\boldsymbol{x}_i) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} + \boldsymbol{\Sigma})$) is given by the scaled leading eigenvectors of the sample covariance matrix $\boldsymbol{C}_{xx} = 1/N \sum_{i=1}^{N} \boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}$, which form the columns of $\boldsymbol{B}$ and the mean of residual (unused) eigenvalues determines $\boldsymbol{\Sigma}$. Thus, in some special cases the solution of the estimation of the FA model may be expressed analytically. Such an approach was denoted as Probabilistic Principal Component Analysis (PPCA), and it is worth writing down the solution

$$\boldsymbol{B} = \boldsymbol{U}(\boldsymbol{\Lambda} - \sigma^2\boldsymbol{I})^{1/2}\boldsymbol{R}, \tag{6.23}$$

where $\boldsymbol{U}$, $\boldsymbol{\Lambda}$ are the matrices of eigenvectors (stored in columns) and of corresponding D_y leading eigenvalues λ_i of the sample covariance matrix $\boldsymbol{C}_{xx}$, respectively, and $\boldsymbol{R}$ is an arbitrary rotation matrix. In this scenario the latent covariance becomes (choosing $\boldsymbol{R} = \boldsymbol{I}$)

$$\boldsymbol{C}_{yy|x} = (\boldsymbol{B}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{B} + \boldsymbol{I})^{-1} = (\sigma^{-2}\left(\boldsymbol{\Lambda} - \sigma^2\boldsymbol{I}\right) + \boldsymbol{I})^{-1} = \sigma^2\boldsymbol{\Lambda}^{-1}.$$

Hence the latent covariance $\boldsymbol{C}_{yy|x}$ is diagonal with diagonal elements σ^2/λ_i. It is obvious that the latent covariance will grow when the noise variance σ^2 is high, but it will grow also in directions where *the data variance is low*. Loosely speaking, the estimate of the latent variable will be more reliable in those subspaces, in which the spread of feature vectors $\boldsymbol{x}_i$ is wider, see Figure 6.1. The rationale are that such a direction can be estimated more reliably than a direction where feature vectors are concentrated near the origin leaving that direction ambiguous.

Another interesting property is that if the noise covariance $\boldsymbol{\Sigma}$ is fixed (e.g. it is derived a-priori, and it is not updated in each iteration – see Section 6.3), the standard FA decomposition described in this section leads to the same result as the FA decomposition of the dataset $\hat{\boldsymbol{X}} = \{\boldsymbol{\Sigma}^{-1/2}\boldsymbol{x}_i\}_{i=1}^N$ with

$$\hat{\boldsymbol{C}}_{yy|x} = (\hat{\boldsymbol{B}}^{\mathrm{T}}\hat{\boldsymbol{B}} + \boldsymbol{I})^{-1} \text{ and } \hat{E}_y[\boldsymbol{y}_i|\boldsymbol{x}_i] = \hat{\boldsymbol{C}}_{yy|x}\hat{\boldsymbol{B}}^{\mathrm{T}}\hat{\boldsymbol{x}}_i, \tag{6.24}$$

up to some rotation matrix $\boldsymbol{R}$, yielding $\boldsymbol{B}\boldsymbol{R} = \boldsymbol{\Sigma}^{1/2}\hat{\boldsymbol{B}}$, $\hat{\boldsymbol{C}}_{yy|x} = \boldsymbol{R}^{\mathrm{T}}\boldsymbol{C}_{yy|x}\boldsymbol{R}$, and $\hat{E}_y[\boldsymbol{y}_i|\boldsymbol{x}_i] = \boldsymbol{R}^{\mathrm{T}}E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]$. It is easy to see that

$$\boldsymbol{\Sigma}^{-1/2}\boldsymbol{x}_i = \hat{\boldsymbol{x}}_i = \hat{\boldsymbol{B}}\hat{E}_y[\boldsymbol{y}_i|\boldsymbol{x}_i] = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{B}\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}\boldsymbol{C}_{yy|x}\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\Sigma}^{-1/2}\boldsymbol{x}_i = E_y[\boldsymbol{y}_i|\boldsymbol{x}_i] =$$
$$= \boldsymbol{\Sigma}^{-1/2}\boldsymbol{B}\boldsymbol{C}_{yy|x}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_i = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{B}E_y[\boldsymbol{y}_i|\boldsymbol{x}_i], \text{ hence } \boldsymbol{x}_i = \boldsymbol{B}E_y[\boldsymbol{y}_i|\boldsymbol{x}_i].$$

Such an observation may bring some additional computational savings when implementing a FA system. The input dataset is normalized beforehand so that all the multiplications with $\boldsymbol{\Sigma}^{-1}$ in the FA update formulas will vanish.

Be aware that we are unable to recover the latent representation $\boldsymbol{y}_i$ of a given feature vector $\boldsymbol{x}_i$, we can recover only the distribution of $\boldsymbol{y}_i$ given $\boldsymbol{x}_i$ and if needed we can use the mean value $\boldsymbol{y}_i^{\mathrm{MAP}} = E_y[\boldsymbol{y}_i|\boldsymbol{x}_i]$ of the distribution as a Maximum A-Posteriory (MAP) point estimate of $\boldsymbol{y}_i$.

6.2 Probabilistic Linear Discriminant Analysis (PLDA)

PLDA was designed as a probabilistic alternative to Linear Discriminant Analysis (LDA), where the aim is to maximize the ratio of between- to within-class covariance in order to increase the separability of given classes. It was introduced to the image processing by Prince and Elder. This section will contain a thorough analysis of the method and its modifications, which will result in a much faster implementation of the training procedure.

Following notations in [8] the PLDA generative model can be written in the form

$$\boldsymbol{x}_{ij} = \boldsymbol{\mu} + \boldsymbol{F}\boldsymbol{h}_i + \boldsymbol{G}\boldsymbol{w}_{ij} + \boldsymbol{\epsilon}_{ij}, \tag{6.25}$$

where vectors from the set $\boldsymbol{X} = \{\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{iJ_i}\}_{i=1}^I, \sum_{i=1}^I J_i = N$ represent I individuals and J_i distinct representations of each individual and N is the number of vectors in $\boldsymbol{X}$. Let us denote the set of distinct representations of one individual as $\boldsymbol{\Lambda}_i = \{\boldsymbol{x}_{ij}\}_{j=1}^{J_i}$. The term $\boldsymbol{\mu} = E[\boldsymbol{x}_{ij}]$ is the mean value of vectors in $\boldsymbol{X}$, $\boldsymbol{h}_i$ is a vector representing the mutual information shared between vectors in $\boldsymbol{\Lambda}_i$, $\boldsymbol{w}_{ij}$ denotes the representation dependent part of each vector in $\boldsymbol{X}$ and $\boldsymbol{\epsilon}_{ij}$ stands for the residual noise factor. Further, columns of the matrix $\boldsymbol{F}$ span the between-individual subspace and columns of the matrix $\boldsymbol{G}$ span the within-individual subspace of the space formed by vectors in $\boldsymbol{X}$. Thus, one can identify the identity component $\boldsymbol{\mu} + \boldsymbol{F}\boldsymbol{h}_i$ and the noise/channel component $\boldsymbol{G}\boldsymbol{w}_{ij} + \boldsymbol{\epsilon}_{ij}$ of each vector $\boldsymbol{x}_{ij}$. Usually dimensions of $\boldsymbol{h}_i$ and $\boldsymbol{w}_{ij}$ are lower than the dimension of the feature vector $\boldsymbol{x}_{ij}$. Denoting $D_x = \dim(\boldsymbol{x}_{ij}) = \dim(\boldsymbol{\epsilon}_{ij})$, $D_h = \dim(\boldsymbol{h}_i)$, $D_w = \dim(\boldsymbol{w}_{ij})$, a reasonable choice would be $D_x > D_h + D_w$. Hence, the high-dimensional representation $\boldsymbol{x}_{ij}$ can be fully explained by two low-dimensional decompositions and some additional information redundancy (caused by the inequality sign). However, the dimensions D_h and D_w are not restricted in any way. For completeness lets add that $\boldsymbol{F}$ is a $D_x \times D_h$ matrix and $\boldsymbol{G}$ is of size $D_x \times D_w$.

In order to facilitate the estimation of unknown parameters restrictions on distributions of $\boldsymbol{h}_i, \boldsymbol{w}_{ij}, \boldsymbol{\epsilon}_{ij}$ are laid as in Section 6.1. We will assume that

$$\boldsymbol{h}_i, \boldsymbol{w}_{ij}, \boldsymbol{\epsilon}_{ij} \text{ are independent and identically distributed (iid)}, \tag{6.26}$$
$$\boldsymbol{h}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \boldsymbol{w}_{i,j} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}), \tag{6.27}$$
$$\boldsymbol{x}_{ij} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{F}\boldsymbol{F}^{\mathrm{T}} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}} + \boldsymbol{\Sigma}), \text{ alternatively } \boldsymbol{x}_{ij}|\boldsymbol{h}_i, \boldsymbol{w}_{ij} \sim \mathcal{N}(\boldsymbol{\mu} + \boldsymbol{F}\boldsymbol{h}_i + \boldsymbol{G}\boldsymbol{w}_{ij}, \boldsymbol{\Sigma}). \tag{6.28}$$

The term $\boldsymbol{I}$ denotes the identity matrix of corresponding dimension. The $D_x \times D_x$ covariance matrix $\boldsymbol{\Sigma}$ of the noise term $\boldsymbol{\epsilon}_{ij}$ will be assumed diagonal until stated otherwise. From (6.28) it is clear that the task of PLDA is to divide the covariance matrix $\boldsymbol{C}_{xx} = 1/N \sum_i \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}}$ of observations $\boldsymbol{x}_i$ into three distinct matrices giving $\boldsymbol{C}_{xx} \approx \boldsymbol{F}\boldsymbol{F}^{\mathrm{T}} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}} + \boldsymbol{\Sigma}$ (equality would be obtained when number of parameters on both sides would equal, but in general this is not the case). The covariance $\boldsymbol{G}\boldsymbol{G}^{\mathrm{T}} + \boldsymbol{\Sigma}$ is responsible for the explanation of the variation of noise. However the advantage in introducing $\boldsymbol{G}$ instead of a full covariance matrix $\boldsymbol{\Sigma}$ and reducing the model into the form $\boldsymbol{x}_{ij} = \boldsymbol{F}\boldsymbol{h}_i + \boldsymbol{\epsilon}_{ij}, \boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\text{full}})$ lays in the number of free parameters of the noise covariance to be estimated. In the reduced form and full covariance case $D_x(D_x+1)/2$ parameters have to be estimated, while $D_x(D_w+1)$ parameters have to be estimated when introducing the matrix $\boldsymbol{G}$ (equality obtained for $D_w = (D_x - 1)/2$). Thus, one can directly control the description power of the noise component.

In order to express the probability distribution $p(\boldsymbol{h}_i, \boldsymbol{w}_{ij}|\boldsymbol{x}_{ij})$, which is again Gaussian, we will rewrite (6.25) into the reduced form

$$\boldsymbol{x}_{ij} = \boldsymbol{B}\boldsymbol{y}_{ij} + \boldsymbol{\epsilon}_{ij}, \boldsymbol{B} = [\boldsymbol{F}, \boldsymbol{G}], \boldsymbol{y}_{ij} = [\boldsymbol{h}_i^{\mathrm{T}}, \boldsymbol{w}_{ij}^{\mathrm{T}}]^{\mathrm{T}}, \tag{6.29}$$

where $\boldsymbol{x}_{ij}$ are now assumed to have zero mean ($\boldsymbol{\mu}$ was subtracted from vectors in $\boldsymbol{X}$ in advance). Since the formula is exactly the same as (6.1), the parameters of the distribution $p(\boldsymbol{y}_{ij}|\boldsymbol{x}_{ij}) = p(\boldsymbol{h}_i, \boldsymbol{w}_{ij}|\boldsymbol{x}_{ij})$ are given as in (6.6) and (6.7).

6.2.1 Training

To train the unknown parameters $\Theta = \{\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{\Sigma}\}$ one has to compute the likelihood that all the vectors in $\boldsymbol{\Lambda}_i$ share the same identity expressed by the latent variable $\boldsymbol{h}_i$. In [8] it was shown how to solve the problem by forming a system of equations for all vectors in $\boldsymbol{\Lambda}_i$ receiving

$$\begin{bmatrix} \boldsymbol{x}_{i1} \\ \boldsymbol{x}_{i2} \\ \vdots \\ \boldsymbol{x}_{iJ_i} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F} & \boldsymbol{G} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{F} & \boldsymbol{0} & \boldsymbol{G} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{F} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{G} \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_i \\ \boldsymbol{w}_{i1} \\ \boldsymbol{w}_{i2} \\ \vdots \\ \boldsymbol{w}_{iJ_i} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \\ \vdots \\ \boldsymbol{\epsilon}_{J_i} \end{bmatrix}, \hat{\boldsymbol{\Sigma}}_i = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{\Sigma} \end{bmatrix}, \tag{6.30}$$

what can be written in a more compact form as

$$\hat{\boldsymbol{x}}_i = \boldsymbol{A}_i \hat{\boldsymbol{y}}_i + \hat{\boldsymbol{\epsilon}}, \tag{6.31}$$

where $\hat{\boldsymbol{\epsilon}} \sim \mathcal{N}(\boldsymbol{0}, \hat{\boldsymbol{\Sigma}}_i)$. Matrices $\boldsymbol{A}_i, \hat{\boldsymbol{\Sigma}}_i$ depend on i through the number of their row- and column-blocks, which are given by the number of vectors in $\boldsymbol{\Lambda}_i$. The joint probability of vectors in $\boldsymbol{\Lambda}_i$ is now equal to

$$p(\boldsymbol{\Lambda}_i) = \mathcal{N}(\hat{\boldsymbol{x}}_i|\boldsymbol{0}, \boldsymbol{A}_i\boldsymbol{A}_i^{\mathrm{T}} + \hat{\boldsymbol{\Sigma}}_i). \tag{6.32}$$

The equation (6.31) is a standard FA problem, which can be solved according to the procedure described in Section 6.1. At first, vector $E_{\hat{y}}[\hat{\boldsymbol{y}}_i|\hat{\boldsymbol{x}}_i]$ is estimated utilizing the matrices $\boldsymbol{A}_i, \hat{\boldsymbol{\Sigma}}_i$ according to (6.7), which is then decomposed to J_i vectors in order to get pairs $(\boldsymbol{x}_{ij}, E_y[\boldsymbol{y}_{ij}|\boldsymbol{x}_{ij}])$ with $\boldsymbol{y}_{ij}$ given in (6.29). Now the reduced problem (6.29) is solved, where $\boldsymbol{B} = [\boldsymbol{F}, \boldsymbol{G}]$. Note that the matrix $\boldsymbol{A}_i$ is used only to get the latent representations $\boldsymbol{h}_i$, $\boldsymbol{w}_{ij}$. The procedure iterates until the convergence or enough iterations have been reached. The complete algorithm is given in Alg.2. Note that $E_{\hat{y}}[\hat{\boldsymbol{y}}_i|\hat{\boldsymbol{x}}_i] = [\bar{\boldsymbol{h}}_i, \bar{\boldsymbol{w}}_{i1}, \dots, \bar{\boldsymbol{w}}_{iJ_i}]^{\mathrm{T}}$, where the bar over the latent variables $\bar{\boldsymbol{h}}_i$, $\bar{\boldsymbol{w}}_{ij}$ denotes that only the MAP estimates (i.e. mean values of $\boldsymbol{h}_i|\boldsymbol{x}_{ij}$, $\boldsymbol{w}_{ij}|\boldsymbol{x}_{ij}$) are computed. Also note that $\boldsymbol{A}_i$, $\hat{\boldsymbol{\Sigma}}_i$ have to be reassembled only when J_i changes, hence the matrix $(\boldsymbol{A}_i^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_i^{-1}\boldsymbol{A}_i + \boldsymbol{I})^{-1}\boldsymbol{A}_i^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_i^{-1}$ is computed only once for each distinct value of J_i (e.g. if J_i is the same for all $\boldsymbol{\Lambda}_i$ then $\boldsymbol{A}_i$, $\hat{\boldsymbol{\Sigma}}_i$ are assembled only once and only one inversion is performed in each iteration in order to compute $E_{\hat{y}}[\hat{\boldsymbol{y}}_i|\hat{\boldsymbol{x}}_i]$).

Algorithm 2 PLDA estimation algorithm

Require: I sets $\boldsymbol{\Lambda}_i = \{\boldsymbol{x}_{ij}\}_{i=1}^{J_i}, i = 1, \ldots, I, \sum_{i=1}^{I} J_i = N$ of distinct representations of an individual i; dimensions D_h, D_w of latent variables $\boldsymbol{h}_i, \boldsymbol{w}_{ij}$; initialization (random) matrices $\boldsymbol{F}_{\text{init}}, \boldsymbol{G}_{\text{init}}, \boldsymbol{\Sigma}_{\text{init}}$; number of iterations N_{iter}

// Initialize:

$\boldsymbol{F} = \boldsymbol{F}_{\text{init}}, \boldsymbol{G} = \boldsymbol{G}_{\text{init}}, \boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\text{init}}$

$\boldsymbol{Z}_{xy} = \boldsymbol{Z}_{yy} = \boldsymbol{0}$

$\boldsymbol{C}_X = \frac{1}{N} \sum_{i,j}^{N} \boldsymbol{x}_{ij} \boldsymbol{x}_{ij}^{\mathrm{T}}$

for $iter = 1$ to N_{iter} **do**

  // Expectation step:

  $\boldsymbol{B} = [\boldsymbol{F}, \boldsymbol{G}]$

  $\boldsymbol{C}_{yy|x} = (\boldsymbol{B}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{B} + \boldsymbol{I})^{-1}$

  for each $\boldsymbol{\Lambda}_i = \{\boldsymbol{x}_{ij}\}_{i=1}^{J_i}, i = 1, \ldots, I$ **do**

    $\hat{\boldsymbol{x}}_i = [\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \ldots, \boldsymbol{x}_{iJ_i}]^{\mathrm{T}}$

    assemble $\boldsymbol{A}_i, \hat{\boldsymbol{\Sigma}}_i$ according to (6.30)

    $E_{\hat{y}}[\hat{\boldsymbol{y}}_i | \hat{\boldsymbol{x}}_i] = (\boldsymbol{A}_i^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_i^{-1} \boldsymbol{A}_i + \boldsymbol{I})^{-1} \boldsymbol{A}_i^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_i^{-1} \hat{\boldsymbol{x}}_i = [\bar{\boldsymbol{h}}_i, \bar{\boldsymbol{w}}_{i1}, \ldots, \bar{\boldsymbol{w}}_{iJ_i}]^{\mathrm{T}}$

    for $j = 1$ to J_i **do**

      $E_y[\boldsymbol{y}_{ij} | \boldsymbol{x}_{ij}] = [\bar{\boldsymbol{h}}_i, \bar{\boldsymbol{w}}_{ij}]^{\mathrm{T}}$ // note: $\boldsymbol{y}_{ij} = [\boldsymbol{h}_i, \boldsymbol{w}_{ij}]^{\mathrm{T}}$

      $E_y[\boldsymbol{y}_{ij} \boldsymbol{y}_{ij}^{\mathrm{T}} | \boldsymbol{x}_{ij}] = \boldsymbol{C}_{yy|x} + E_y[\boldsymbol{y}_{ij} | \boldsymbol{x}_{ij}] E_y[\boldsymbol{y}_{ij}^{\mathrm{T}} | \boldsymbol{x}_{ij}]$

      $\boldsymbol{Z}_{xy} = \boldsymbol{Z}_{xy} + \boldsymbol{x}_{ij} E_y[\boldsymbol{y}_{ij}^{\mathrm{T}} | \boldsymbol{x}_{ij}]$

      $\boldsymbol{Z}_{yy} = \boldsymbol{Z}_{yy} + E_y[\boldsymbol{y}_{ij} \boldsymbol{y}_{ij}^{\mathrm{T}} | \boldsymbol{x}_{ij}]$

    end for

  end for

  // Maximization step:

  $\boldsymbol{B} = \boldsymbol{Z}_{xy} \boldsymbol{Z}_{yy}^{-1}$

  $\boldsymbol{\Sigma} = \operatorname{diagz}\left(\boldsymbol{C}_X - \frac{1}{N} \boldsymbol{B} \boldsymbol{Z}_{xy}^{\mathrm{T}}\right)$

  decompose $\boldsymbol{B}$ to $\boldsymbol{F}$ and $\boldsymbol{G}$

end for

return $\boldsymbol{F}$, $\boldsymbol{G}$ and $\boldsymbol{\Sigma}$

6.2.2 Training Revisited I

The problem associated with the training procedure described in the previous section is that the matrix $(\boldsymbol{A}_i^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_i^{-1} \boldsymbol{A}_i + \boldsymbol{I})^{-1}$ given in (6.6) has to be computed in each training iteration. Hence, since the size and the structure of $\boldsymbol{A}_i$ depends on the number of vectors in $\boldsymbol{\Lambda}_i$, it has to be reassembled and multiplied whenever the number of vectors in $\boldsymbol{\Lambda}_i$ changes, see (6.30). And moreover, the huge matrix $(\boldsymbol{A}_i^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_i^{-1} \boldsymbol{A}_i + \boldsymbol{I})$ has to be inverted in order to evaluate (6.7). When observations $\boldsymbol{x}_{ij}$ are of higher dimension (this is the case when working with supervectors) and/or the number of representations of an individual is high, the inversion of the matrix $(\boldsymbol{A}_i^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_i^{-1} \boldsymbol{A}_i + \boldsymbol{I})$ can easily become intractable. Even if a reliable inversion can be computed, the memory management will become expensive. In this and the following subsection procedures are going to be proposed in order to facilitate the estimation process. We will show how to invert the matrix given in (6.6) leading to a much faster and easier implementation. Also, we are going to analyze the solution and infer some conclusions concerning the algorithm of PLDA.

The task is to compute $\boldsymbol{C}_{\hat{y}\hat{y}|\hat{x}} = (\boldsymbol{A}_i^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_i^{-1} \boldsymbol{A}_i + \boldsymbol{I})^{-1}$ in order to evaluate (6.7), thus to invert a

huge matrix $\boldsymbol{A}_i^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_i^{-1}\boldsymbol{A}_i + \boldsymbol{I}$. According to (6.30) (because of lucidity the case for $J_i = 3$ is given)

$$\begin{aligned}\boldsymbol{C}_{\hat{y}\hat{y}|\hat{x}}^{-1} &= \begin{bmatrix} \boldsymbol{F}^{\mathrm{T}} & \boldsymbol{F}^{\mathrm{T}} & \boldsymbol{F}^{\mathrm{T}} \\ \boldsymbol{G}^{\mathrm{T}} & 0 & 0 \\ 0 & \boldsymbol{G}^{\mathrm{T}} & 0 \\ 0 & 0 & \boldsymbol{G}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}^{-1} & 0 & 0 \\ 0 & \boldsymbol{\Sigma}^{-1} & 0 \\ 0 & 0 & \boldsymbol{\Sigma}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{F} & \boldsymbol{G} & 0 & 0 \\ \boldsymbol{F} & 0 & \boldsymbol{G} & 0 \\ \boldsymbol{F} & 0 & 0 & \boldsymbol{G} \end{bmatrix} + \begin{bmatrix} \boldsymbol{I} & 0 & 0 & 0 \\ 0 & \boldsymbol{I} & 0 & 0 \\ 0 & 0 & \boldsymbol{I} & 0 \\ 0 & 0 & 0 & \boldsymbol{I} \end{bmatrix} \\ &= \begin{bmatrix} 3\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{F} + \boldsymbol{I} & \boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} & \boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} & \boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} \\ \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{F} & \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I} & 0 & 0 \\ \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{F} & 0 & \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I} & 0 \\ \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{F} & 0 & 0 & \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I} \end{bmatrix}. \end{aligned} \tag{6.33}$$

This can be written in a block form as

$$\boldsymbol{C}_{\hat{y}\hat{y}|\hat{x}} = \begin{bmatrix} \boldsymbol{\Omega}_1 & \boldsymbol{\Omega}_3 \\ \boldsymbol{\Omega}_3^{\mathrm{T}} & \boldsymbol{\Omega}_2 \end{bmatrix}^{-1} = \begin{bmatrix} \tilde{\boldsymbol{\Omega}}_1 & \tilde{\boldsymbol{\Omega}}_3 \\ \tilde{\boldsymbol{\Omega}}_3^{\mathrm{T}} & \tilde{\boldsymbol{\Omega}}_2 \end{bmatrix}, \tag{6.34}$$

where the decomposition of the matrix $\boldsymbol{C}_{\hat{y}\hat{y}|\hat{x}}^{-1}$ can be suitably chosen as

$$\boldsymbol{\Omega}_1 = J_i\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{F} + \boldsymbol{I}, \qquad (D_h \times D_h), \tag{6.35}$$

$$\boldsymbol{\Omega}_2 = \begin{bmatrix} \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I} & \boldsymbol{0} & \vdots \\ \vdots & \boldsymbol{0} & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I} \end{bmatrix}, \quad (J_iD_w \times J_iD_w), \tag{6.36}$$

$$\boldsymbol{\Omega}_3 = \begin{bmatrix} \boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G}, & \boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G}, & \dots, & \boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} \end{bmatrix}, \qquad (D_h \times J_iD_w), \tag{6.37}$$

where sizes of individual matrices are given in brackets and J_i is the number of vectors in $\boldsymbol{\Lambda}_i$. Following the formulas for block inverses

$$\begin{aligned} \tilde{\boldsymbol{\Omega}}_1 &= (\boldsymbol{\Omega}_1 - \boldsymbol{\Omega}_3\boldsymbol{\Omega}_2^{-1}\boldsymbol{\Omega}_3^{\mathrm{T}})^{-1}, \\ \tilde{\boldsymbol{\Omega}}_3 &= -\tilde{\boldsymbol{\Omega}}_1\boldsymbol{\Omega}_3\boldsymbol{\Omega}_2^{-1}, \\ \tilde{\boldsymbol{\Omega}}_2 &= \boldsymbol{\Omega}_2^{-1} + \boldsymbol{\Omega}_2^{-1}\boldsymbol{\Omega}_3^{\mathrm{T}}\tilde{\boldsymbol{\Omega}}_1\boldsymbol{\Omega}_3\boldsymbol{\Omega}_2^{-1}. \end{aligned} \tag{6.38}$$

we can note that the sizes of corresponding blocks (e.g. $\tilde{\boldsymbol{\Omega}}_3$ and $\boldsymbol{\Omega}_3$) remain the same. Since blocks in $\boldsymbol{\Omega}_3$ repeat, we can state that $\tilde{\boldsymbol{\Omega}}_3$ will contain J_i identical block-matrices $\tilde{\boldsymbol{\Omega}}_{3\mathrm{S}}$, each of size $D_h \times D_w$, and since $\boldsymbol{\Omega}_3^{\mathrm{T}}\tilde{\boldsymbol{\Omega}}_1\boldsymbol{\Omega}_3$ is full and contains identical $D_w \times D_w$ blocks and $\boldsymbol{\Omega}_2$ is block-diagonal, $\tilde{\boldsymbol{\Omega}}_2$ will contain two repeating block-matrices, where $\tilde{\boldsymbol{\Omega}}_{2\mathrm{D}}$ will be repeated on the diagonal and $\tilde{\boldsymbol{\Omega}}_{2\mathrm{S}}$ will fill the non-diagonal blocks (for more details see [72]). Hence, when estimating (6.7) we get (again, because of lucidity only the case for $J_i = 3$ is given)

$$\begin{aligned} E_y[\hat{\boldsymbol{y}}_i|\hat{\boldsymbol{x}}_i] = \boldsymbol{C}_{\hat{y}\hat{y}|\hat{x}}\boldsymbol{A}_i^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_i^{-1}\hat{\boldsymbol{x}}_i = (\boldsymbol{A}_i^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_i^{-1}\boldsymbol{A}_i^{\mathrm{T}} + \boldsymbol{I})^{-1}\boldsymbol{A}_i^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_i^{-1}\hat{\boldsymbol{x}}_i = \\ \begin{bmatrix} \bar{\boldsymbol{h}}_i \\ \bar{\boldsymbol{w}}_{i1} \\ \bar{\boldsymbol{w}}_{i2} \\ \bar{\boldsymbol{w}}_{i3} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{\Omega}}_1 & \tilde{\boldsymbol{\Omega}}_{3\mathrm{S}} & \tilde{\boldsymbol{\Omega}}_{3\mathrm{S}} & \tilde{\boldsymbol{\Omega}}_{3\mathrm{S}} \\ \tilde{\boldsymbol{\Omega}}_{3\mathrm{S}}^{\mathrm{T}} & \tilde{\boldsymbol{\Omega}}_{2\mathrm{D}} & \tilde{\boldsymbol{\Omega}}_{2\mathrm{S}} & \tilde{\boldsymbol{\Omega}}_{2\mathrm{S}} \\ \tilde{\boldsymbol{\Omega}}_{3\mathrm{S}}^{\mathrm{T}} & \tilde{\boldsymbol{\Omega}}_{2\mathrm{S}} & \tilde{\boldsymbol{\Omega}}_{2\mathrm{D}} & \tilde{\boldsymbol{\Omega}}_{2\mathrm{S}} \\ \tilde{\boldsymbol{\Omega}}_{3\mathrm{S}}^{\mathrm{T}} & \tilde{\boldsymbol{\Omega}}_{2\mathrm{S}} & \tilde{\boldsymbol{\Omega}}_{2\mathrm{S}} & \tilde{\boldsymbol{\Omega}}_{2\mathrm{D}} \end{bmatrix} \begin{bmatrix} \boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\sum_{j=1}^{3}\boldsymbol{x}_{ij} \\ \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_{i1} \\ \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_{i2} \\ \boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_{i3} \end{bmatrix}, \end{aligned} \tag{6.39}$$

where $\bar{\boldsymbol{h}}_i = E_h[\boldsymbol{h}_i|\hat{\boldsymbol{x}}_i], \bar{\boldsymbol{w}}_{ij} = E_w[\boldsymbol{w}_{ij}|\hat{\boldsymbol{x}}_i]$. Note that since $\hat{\boldsymbol{y}}_i^{\mathrm{T}} = [\boldsymbol{h}_i^{\mathrm{T}}, \boldsymbol{w}_{i1}^{\mathrm{T}}, \dots, \boldsymbol{w}_{iJ_i}^{\mathrm{T}}]$ we get

$$\boldsymbol{C}_{hh|\hat{x}} = \tilde{\boldsymbol{\Omega}}_1, \ \boldsymbol{C}_{w_iw_i|\hat{x}} = \tilde{\boldsymbol{\Omega}}_{2\mathrm{D}}, \ \boldsymbol{C}_{hw|\hat{x}} = \tilde{\boldsymbol{\Omega}}_{3\mathrm{S}}, \ \boldsymbol{C}_{w_iw_j|\hat{x}} = \tilde{\boldsymbol{\Omega}}_{2\mathrm{S}}, \tag{6.40}$$

where $\boldsymbol{C}_{hw|\hat{x}}$ is the covariance of latent variables $\boldsymbol{h}$ and $\boldsymbol{w}$ given an observation vector $\hat{\boldsymbol{x}}$, etc. In order to estimate the inverses we will use inverse identities given in Appendix C. Now we are ready to plug (6.35), (6.36), (6.37) into (6.38) to get

$$\begin{aligned}
\tilde{\boldsymbol{\Omega}}_1 &= \left(\boldsymbol{I} + J_i \boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{F} - J_i \boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G}(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I})^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{F}\right)^{-1} = \\
&= \frac{1}{J_i}\left(\boldsymbol{F}^{\mathrm{T}}(\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}\boldsymbol{F} + \frac{1}{J_i}\boldsymbol{I}\right)^{-1}, \qquad (6.41) \\
\tilde{\boldsymbol{\Omega}}_{2\mathrm{S}} &= (\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I})^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{F}\tilde{\boldsymbol{\Omega}}_1\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G}(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I})^{-1} = \\
&= \boldsymbol{G}^{\mathrm{T}}(\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}\boldsymbol{F}\tilde{\boldsymbol{\Omega}}_1\boldsymbol{F}^{\mathrm{T}}(\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}\boldsymbol{G}, \qquad (6.42) \\
\tilde{\boldsymbol{\Omega}}_{2\mathrm{D}} &= \tilde{\boldsymbol{\Omega}}_{2\mathrm{S}} + (\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I})^{-1}, \qquad (6.43) \\
\tilde{\boldsymbol{\Omega}}_{3\mathrm{S}} &= -\tilde{\boldsymbol{\Omega}}_1\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G}(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I})^{-1} = -\tilde{\boldsymbol{\Omega}}_1\boldsymbol{F}^{\mathrm{T}}(\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}\boldsymbol{G}. \qquad (6.44)
\end{aligned}$$

And utilizing (6.41)-(6.44) and (6.39) we can derive the formulas for $\bar{\boldsymbol{h}}_i$ and $\bar{\boldsymbol{w}}_{ij}$, where

$$\begin{aligned}
\bar{\boldsymbol{h}}_i &= \tilde{\boldsymbol{\Omega}}_1\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\sum_{j=1}^{J_i}\boldsymbol{x}_{ij} + \tilde{\boldsymbol{\Omega}}_{3\mathrm{S}}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\sum_{j=1}^{J_i}\boldsymbol{x}_{ij} = \\
&= \tilde{\boldsymbol{\Omega}}_1\boldsymbol{F}^{\mathrm{T}}\left(\boldsymbol{\Sigma}^{-1} - (\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}\boldsymbol{G}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\right)\sum_{j=1}^{J_i}\boldsymbol{x}_{ij} = \\
&= \left(\boldsymbol{F}^{\mathrm{T}}(\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}\boldsymbol{F} + \frac{1}{J_i}\boldsymbol{I}\right)^{-1}\boldsymbol{F}^{\mathrm{T}}(\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}\frac{1}{J_i}\sum_{j=1}^{J_i}\boldsymbol{x}_{ij}, \qquad (6.45)
\end{aligned}$$

and in analogy with previous steps we get

$$\begin{aligned}
\bar{\boldsymbol{w}}_{ij} &= \tilde{\boldsymbol{\Omega}}_{3\mathrm{S}}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\sum_{j=1}^{J_i}\boldsymbol{x}_{ij} + \tilde{\boldsymbol{\Omega}}_{2\mathrm{D}}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_{ij} + \tilde{\boldsymbol{\Omega}}_{2\mathrm{S}}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\sum_{k=1,k\neq j}^{J_i}\boldsymbol{x}_{ik} = \\
&= \tilde{\boldsymbol{\Omega}}_{3\mathrm{S}}^{\mathrm{T}}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\sum_{j=1}^{J_i}\boldsymbol{x}_{ij} + \tilde{\boldsymbol{\Omega}}_{2\mathrm{S}}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\sum_{j=1}^{J_i}\boldsymbol{x}_{ij} + (\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I})^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_{ij} = \\
&= \boldsymbol{G}^{\mathrm{T}}(\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}\left[-\boldsymbol{F}\tilde{\boldsymbol{\Omega}}_1\boldsymbol{F}^{\mathrm{T}}\left(\boldsymbol{\Sigma}^{-1} - (\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}\boldsymbol{G}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\right)\sum_{j=1}^{J_i}\boldsymbol{x}_{ij} + \boldsymbol{x}_{ij}\right] = \\
&= \boldsymbol{G}^{\mathrm{T}}(\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}\left[\boldsymbol{x}_{ij} - \boldsymbol{F}\left(\tilde{\boldsymbol{\Omega}}_1\boldsymbol{F}^{\mathrm{T}}(\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}\sum_{j=1}^{J_i}\boldsymbol{x}_{ij}\right)\right] = \\
&= \left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I}\right)^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}_{ij} - \boldsymbol{F}\bar{\boldsymbol{h}}_i\right). \qquad (6.46)
\end{aligned}$$

In summary, we derived formulas

$$\begin{aligned}
\boldsymbol{\Sigma}_{\mathrm{N}} &= \boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}}, \boldsymbol{\mu}_i = \frac{1}{J_i}\sum_{j=1}^{J_i}\boldsymbol{x}_{ij}, \qquad (6.47) \\
\bar{\boldsymbol{h}}_i &= \left(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathrm{N}}^{-1}\boldsymbol{F} + \frac{1}{J_i}\boldsymbol{I}\right)^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathrm{N}}^{-1}\boldsymbol{\mu}_i, \qquad (6.48) \\
\bar{\boldsymbol{w}}_{ij} &= \left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I}\right)^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}_{ij} - \boldsymbol{F}\bar{\boldsymbol{h}}_i\right). \qquad (6.49)
\end{aligned}$$

The latent variable $\bar{\boldsymbol{h}}_i$, which represents the mutual information, is the projected mean of all the given representations of an individual i assuming full noise covariance $\boldsymbol{\Sigma}_{\mathrm{N}}$. In addition, the more representations are given the lesser is the influence of the prior $\boldsymbol{h}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. The representation dependent latent variable $\bar{\boldsymbol{w}}_{ij}$ is the projection of the residual $(\boldsymbol{x}_{ij} - \boldsymbol{F}\bar{\boldsymbol{h}}_i)$ on the space formed by columns of $\boldsymbol{G}$ assuming only the unexplained variance $\boldsymbol{\Sigma}$, however since only one vector is used at a time the prior is fixed. It is also obvious that $\bar{\boldsymbol{h}}_i$ depends not only on one particular $\boldsymbol{x}_{ij} \in \boldsymbol{\Lambda}_i$, but on the whole set $\boldsymbol{\Lambda}_i$, whereas $\bar{\boldsymbol{w}}_{ij}$ depends on $\boldsymbol{x}_{ij} \in \boldsymbol{\Lambda}_i$ and even on $\boldsymbol{h}_i$, i.e. $\bar{\boldsymbol{h}}_i = E_h[\boldsymbol{h}_i|\boldsymbol{\Lambda}_i], \bar{\boldsymbol{w}}_{ij} = E_w[\boldsymbol{w}_{ij}|\boldsymbol{h}_i, \boldsymbol{x}_{ij}]$.

Sizes of the matrices to be inverted are now $D_h \times D_h$ and $D_w \times D_w$ instead of $(D_h + J_i Dw) \times (D_h + J_i Dw)$, which is the size of $\boldsymbol{C}_{\hat{y}\hat{y}|\hat{x}}$. In cases where the number of representations in $\boldsymbol{\Lambda}_i$ is high the term $1/J_i\boldsymbol{I}$ in (6.48) can be left out or set arbitrary small to handle ill-conditioned situations. Thus, one can compute the transformation matrices $\boldsymbol{T}_F = \left(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathrm{N}}^{-1}\boldsymbol{F} + \rho\boldsymbol{I}\right)^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathrm{N}}^{-1}$ (for some small ρ) and $\boldsymbol{T}_G = \left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{I}\right)^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}$ before each EM iteration, and iterate through all the data without the need to reestimate/invert any of the matrices when the amount of data in any of the sets $\boldsymbol{\Lambda}_i$ changes. Hence, the difference from the training procedure described in Section 6.2.1 stands in the use of $\boldsymbol{T}_F$ and $\boldsymbol{T}_G$ instead of $\boldsymbol{A}_i$ when estimating $E_y[\boldsymbol{y}_{ij}|\boldsymbol{x}_{ij}] = [\bar{\boldsymbol{h}}_i^{\mathrm{T}}, \bar{\boldsymbol{w}}_{ij}^{\mathrm{T}}]^{\mathrm{T}}$. The complete estimation procedure is given in Alg.3. Note that $\boldsymbol{C}_{yy|x}$ can be obtained as

$$\boldsymbol{C}_{yy|x} = (\boldsymbol{B}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{B} + \boldsymbol{I})^{-1} = \begin{bmatrix} \boldsymbol{C}_{hh|x} & \boldsymbol{C}_{hw|x} \\ \boldsymbol{C}_{hw|x}^{\mathrm{T}} & \boldsymbol{C}_{w_i w_i|x} \end{bmatrix} \tag{6.50}$$

where $J_i = 1$ since we work with individual feature vectors $\boldsymbol{x}_{ij}$.

6.2.3 Training Revisited II

The goal of the previous section was to facilitate evaluations of latent variables, now we will focus on the accumulation process in the PLDA training, more precisely on individual terms in update formulas (6.17) and (6.20). Utilizing notations from previous section, these equations can be rewritten as

$$\boldsymbol{B} = \left(\sum_{ij}\left[\boldsymbol{x}_{ij}\bar{\boldsymbol{h}}_i^{\mathrm{T}}, \boldsymbol{x}_{ij}\bar{\boldsymbol{w}}_{ij}^{\mathrm{T}}\right]\right)\left(\sum_{ij}\begin{bmatrix} \bar{\boldsymbol{h}}_i\bar{\boldsymbol{h}}_i^{\mathrm{T}} + \boldsymbol{C}_{hh|\hat{x}}, & \bar{\boldsymbol{h}}_i\bar{\boldsymbol{w}}_{ij}^{\mathrm{T}} + \boldsymbol{C}_{hw|\hat{x}} \\ \bar{\boldsymbol{w}}_{ij}\bar{\boldsymbol{h}}_i^{\mathrm{T}} + \boldsymbol{C}_{hw|\hat{x}}^{\mathrm{T}}, & \bar{\boldsymbol{w}}_{ij}\bar{\boldsymbol{w}}_{ij}^{\mathrm{T}} + \boldsymbol{C}_{w_i w_i|\hat{x}} \end{bmatrix}\right)^{-1}, \tag{6.51}$$

$$\boldsymbol{\Sigma} = \mathrm{diagz}\left(\frac{1}{N}\sum_{ij}\boldsymbol{x}_{ij}\boldsymbol{x}_{ij}^{\mathrm{T}} - \frac{1}{N}\sum_{ij}\boldsymbol{F}\bar{\boldsymbol{h}}_i\boldsymbol{x}_{ij}^{\mathrm{T}} - \frac{1}{N}\sum_{ij}\boldsymbol{G}\bar{\boldsymbol{w}}_{ij}\boldsymbol{x}_{ij}^{\mathrm{T}}\right). \tag{6.52}$$

We will now show that in order to update the matrices we do not need the whole data set $\boldsymbol{X}$, but only some of the data covariances. Since PLDA requires several iterations to converge such an approach will significantly speed up the estimation especially when the amount of input vectors is huge. The fact that the training algorithm and dataset depend only on some statistics of the dataset is common in many statistical algorithms (e.g. in techniques from Chapter 3), and it frequently facilitates the estimation procedure.

Firstly, let $\boldsymbol{X} = [\boldsymbol{x}_{11}, \ldots, \boldsymbol{x}_{1J_1}, \ldots, \boldsymbol{x}_{I1}, \ldots, \boldsymbol{x}_{IJ_I}]$ be the matrix of successively ordered input data (as described in the context of (5.7)). Denote

$$\boldsymbol{C}_X = \frac{1}{N}\sum_{i=1}^{I}\sum_{j=1}^{J_i}\boldsymbol{x}_{ij}\boldsymbol{x}_{ij}^{\mathrm{T}} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}, \tag{6.53}$$

$$\boldsymbol{C}_B = \frac{1}{N}\sum_{i=1}^{I}J_i\boldsymbol{\mu}_i\boldsymbol{\mu}_i^{\mathrm{T}} = \frac{1}{N}\boldsymbol{X}\boldsymbol{J}\boldsymbol{X}^{\mathrm{T}}, \tag{6.54}$$

Algorithm 3 PLDA estimation algorithm revisited I

Require: I sets $\boldsymbol{\Lambda}_i = \{\boldsymbol{x}_{ij}\}_{i=1}^{J_i}, i = 1, \ldots, I, \sum_{i=1}^{I} J_i = N$ of distinct representations of an individual i; dimensions D_h, D_w of latent variables $\boldsymbol{h}_i, \boldsymbol{w}_{ij}$; initialization (random) matrices $\boldsymbol{F}_{\text{init}}, \boldsymbol{G}_{\text{init}}, \boldsymbol{\Sigma}_{\text{init}}$; number of iterations N_{iter}; small ρ

// Initialize:

$\boldsymbol{F} = \boldsymbol{F}_{\text{init}}, \boldsymbol{G} = \boldsymbol{G}_{\text{init}}, \boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\text{init}}$

$\boldsymbol{Z}_{xy} = \boldsymbol{Z}_{yy} = \boldsymbol{0}$

$\boldsymbol{C}_X = \frac{1}{N} \sum_{i,j}^{N} \boldsymbol{x}_{ij} \boldsymbol{x}_{ij}^{\mathrm{T}}$

for $iter = 1$ to N_{iter} **do**

  // Expectation step:

  $\boldsymbol{B} = [\boldsymbol{F}, \boldsymbol{G}]$

  $\boldsymbol{C}_{yy|x} = (\boldsymbol{B}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{B} + \boldsymbol{I})^{-1}$

  $\boldsymbol{\Sigma}_N^{-1} = (\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}$

  $\boldsymbol{T}_F = \left(\boldsymbol{F}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathrm{N}}^{-1} \boldsymbol{F} + \rho \boldsymbol{I}\right)^{-1} \boldsymbol{F}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathrm{N}}^{-1}$

  $\boldsymbol{T}_G = \left(\boldsymbol{G}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{G} + \boldsymbol{I}\right)^{-1} \boldsymbol{G}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}$

  for each $\boldsymbol{\Lambda}_i = \{\boldsymbol{x}_{ij}\}_{i=1}^{J_i}, i = 1, \ldots, I$ **do**

    $\bar{\boldsymbol{h}}_i = \boldsymbol{T}_F \boldsymbol{\mu}_i$, $\boldsymbol{\mu}_i = \frac{1}{J_i} \sum_{j=1}^{J_i} \boldsymbol{x}_{ij}$

    for $j = 1$ to J_i **do**

      $\bar{\boldsymbol{w}}_{ij} = \boldsymbol{T}_G(\boldsymbol{x}_{ij} - \boldsymbol{F}\bar{\boldsymbol{h}}_i)$

      $E_y[\boldsymbol{y}_{ij}|\boldsymbol{x}_{ij}] = [\bar{\boldsymbol{h}}_i, \bar{\boldsymbol{w}}_{ij}]^{\mathrm{T}}$ // note: $\boldsymbol{y}_{ij} = [\boldsymbol{h}_i, \boldsymbol{w}_{ij}]^{\mathrm{T}}$

      $E_y[\boldsymbol{y}_{ij}\boldsymbol{y}_{ij}^{\mathrm{T}}|\boldsymbol{x}_{ij}] = \boldsymbol{C}_{yy|x} + E_y[\boldsymbol{y}_{ij}|\boldsymbol{x}_{ij}] E_y[\boldsymbol{y}_{ij}^{\mathrm{T}}|\boldsymbol{x}_{ij}]$

      $\boldsymbol{Z}_{xy} = \boldsymbol{Z}_{xy} + \boldsymbol{x}_{ij} E_y[\boldsymbol{y}_{ij}^{\mathrm{T}}|\boldsymbol{x}_{ij}]$

      $\boldsymbol{Z}_{yy} = \boldsymbol{Z}_{yy} + E_y[\boldsymbol{y}_{ij}\boldsymbol{y}_{ij}^{\mathrm{T}}|\boldsymbol{x}_{ij}]$

    end for

  end for

  // Maximization step:

  $\boldsymbol{B} = \boldsymbol{Z}_{xy} \boldsymbol{Z}_{yy}^{-1}$

  $\boldsymbol{\Sigma} = \text{diagz}\left(\boldsymbol{C}_X - \frac{1}{N} \boldsymbol{B} \boldsymbol{Z}_{xy}^{\mathrm{T}}\right)$

  decompose $\boldsymbol{B}$ to $\boldsymbol{F}$ and $\boldsymbol{G}$

end for

return $\boldsymbol{F}$, $\boldsymbol{G}$ and $\boldsymbol{\Sigma}$

$$\boldsymbol{J} = \begin{bmatrix} \frac{1}{J_1^2}\mathbb{I}_{J_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{1}{J_I^2}\mathbb{I}_{J_I} \end{bmatrix}, \ \boldsymbol{\mu}_i = \frac{1}{J_i} \sum_{j=1}^{J_i} \boldsymbol{x}_{ij},$$

where $\boldsymbol{C}_X$ is the overall data covariance, $\boldsymbol{C}_B$ is the weighted between-class covariance, $\mathbb{I}_{J_i}$ is a $J_i \times J_i$ matrix with 1 in each entry, and

$$\boldsymbol{T}_F = \left(\boldsymbol{F}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathrm{N}}^{-1} \boldsymbol{F} + \rho \boldsymbol{I}\right)^{-1} \boldsymbol{F}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathrm{N}}^{-1}, \tag{6.55}$$

$$\boldsymbol{T}_G = \left(\boldsymbol{G}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{G} + \boldsymbol{I}\right)^{-1} \boldsymbol{G}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}, \tag{6.56}$$

where $\bar{\boldsymbol{h}}_i = \boldsymbol{T}_F \boldsymbol{\mu}_i$ and $\bar{\boldsymbol{w}}_{ij} = \boldsymbol{T}_G \left(\boldsymbol{x}_{ij} - \boldsymbol{F}\bar{\boldsymbol{h}}_i\right)$, and ρ is a small constant arbitrary chosen. Now let us

focus on the individual terms in (6.51), hence

$$\boldsymbol{E}_{xh} = \sum_{ij} \boldsymbol{x}_{ij}\bar{\boldsymbol{h}}_i^{\mathrm{T}} = \left(\sum_{ij} \boldsymbol{x}_{ij}\boldsymbol{\mu}_i^{\mathrm{T}}\right)\boldsymbol{T}_F^{\mathrm{T}} = \left(\sum_{i} J_i\boldsymbol{\mu}_i\boldsymbol{\mu}_i^{\mathrm{T}}\right)\boldsymbol{T}_F^{\mathrm{T}} = N\boldsymbol{C}_B\boldsymbol{T}_F^{\mathrm{T}}, \tag{6.57}$$

$$\boldsymbol{E}_{xw} = \sum_{ij} \boldsymbol{x}_{ij}\bar{\boldsymbol{w}}_{ij}^{\mathrm{T}} = \sum_{ij} \boldsymbol{x}_{ij}(\boldsymbol{x}_{ij} - \boldsymbol{F}\bar{\boldsymbol{h}}_i)^{\mathrm{T}}\boldsymbol{T}_G^{\mathrm{T}} = (N\boldsymbol{C}_X - \boldsymbol{E}_{xh}\boldsymbol{F}^{\mathrm{T}})\boldsymbol{T}_G^{\mathrm{T}} = N\boldsymbol{\Sigma}_{/F}\boldsymbol{T}_G^{\mathrm{T}}, \tag{6.58}$$

where

$$\boldsymbol{\Sigma}_{/F} = \boldsymbol{C}_X - \frac{1}{N}(\boldsymbol{F}\boldsymbol{E}_{xh}^{\mathrm{T}})^{\mathrm{T}} \tag{6.59}$$

is according to (6.18) and (6.20) the residual covariance not captured by $\boldsymbol{F}\boldsymbol{F}^{\mathrm{T}}$ (in fact, this is true only when the convergence was acquired – see the discussion after (6.64)). Further

$$\boldsymbol{E}_{hh} = \sum_{ij} \bar{\boldsymbol{h}}_i\bar{\boldsymbol{h}}_i^{\mathrm{T}} = \sum_{i} J_i\bar{\boldsymbol{h}}_i\bar{\boldsymbol{h}}_i^{\mathrm{T}} = \boldsymbol{T}_F\left(\sum_{i} J_i\boldsymbol{\mu}_i\boldsymbol{\mu}_i^{\mathrm{T}}\right)\boldsymbol{T}_F^{\mathrm{T}} = N\boldsymbol{T}_F\boldsymbol{C}_B\boldsymbol{T}_F^{\mathrm{T}}, \tag{6.60}$$

$$\boldsymbol{E}_{hw} = \sum_{ij} \bar{\boldsymbol{h}}_i\bar{\boldsymbol{w}}_{ij}^{\mathrm{T}} = \sum_{ij} \bar{\boldsymbol{h}}_i(\boldsymbol{x}_{ij} - \boldsymbol{F}\bar{\boldsymbol{h}}_i)^{\mathrm{T}}\boldsymbol{T}_G^{\mathrm{T}} = -(\boldsymbol{F}\boldsymbol{E}_{hh} - \boldsymbol{E}_{xh})^{\mathrm{T}}\boldsymbol{T}_G^{\mathrm{T}}. \tag{6.61}$$

$$\begin{aligned}
\boldsymbol{E}_{ww} &= \sum_{ij} \bar{\boldsymbol{w}}_{ij}\bar{\boldsymbol{w}}_{ij}^{\mathrm{T}} = \boldsymbol{T}_G\sum_{ij}(\boldsymbol{x}_{ij} - \boldsymbol{F}\bar{\boldsymbol{h}}_i)(\boldsymbol{x}_{ij} - \boldsymbol{F}\bar{\boldsymbol{h}}_i)^{\mathrm{T}}\boldsymbol{T}_G^{\mathrm{T}} = \\
&= \boldsymbol{T}_G\left(N\boldsymbol{C}_X - \boldsymbol{E}_{xh}\boldsymbol{F}^{\mathrm{T}} - \boldsymbol{F}\boldsymbol{E}_{xh}^{\mathrm{T}} + \boldsymbol{F}\boldsymbol{E}_{hh}\boldsymbol{F}^{\mathrm{T}}\right)\boldsymbol{T}_G^{\mathrm{T}} = \\
&= N\boldsymbol{T}_G\boldsymbol{\Sigma}_{/F}\boldsymbol{T}_G^{\mathrm{T}} + \boldsymbol{T}_G\boldsymbol{F}(\boldsymbol{F}\boldsymbol{E}_{hh} - \boldsymbol{E}_{xh})^{\mathrm{T}}\boldsymbol{T}_G^{\mathrm{T}} = \\
&= N\boldsymbol{T}_G\boldsymbol{\Sigma}_{/F}\boldsymbol{T}_G^{\mathrm{T}} - \boldsymbol{T}_G\boldsymbol{F}\boldsymbol{E}_{hw} = \boldsymbol{T}_G(\boldsymbol{E}_{xw} - \boldsymbol{F}\boldsymbol{E}_{hw}),
\end{aligned} \tag{6.62}$$

Now the update formulas (6.51) and (6.52) have the form

$$\begin{bmatrix} \boldsymbol{E}_{hh} + N\boldsymbol{C}_{hh|\hat{x}}, & \boldsymbol{E}_{hw} + N\boldsymbol{C}_{hw|\hat{x}} \\ \boldsymbol{E}_{hw}^{\mathrm{T}} + N\boldsymbol{C}_{hw|\hat{x}}^{\mathrm{T}}, & \boldsymbol{E}_{ww} + N\boldsymbol{C}_{w_iw_i|\hat{x}} \end{bmatrix}\begin{bmatrix} \boldsymbol{F}^{\mathrm{T}} \\ \boldsymbol{G}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{E}_{xh}^{\mathrm{T}} \\ \boldsymbol{E}_{xw}^{\mathrm{T}} \end{bmatrix} \tag{6.63}$$

solved for $\boldsymbol{F}$ and $\boldsymbol{G}$, where a system of equation is solved rather than to compute the inverse of a matrix. Let $\boldsymbol{F}^*$ and $\boldsymbol{G}^*$ be the solutions of (6.63). Using the already accumulated $\boldsymbol{E}_{xh}$ and $\boldsymbol{E}_{xw}$ we get

$$\boldsymbol{\Sigma} = \mathrm{diagz}\left(\boldsymbol{C}_X - \frac{1}{N}\boldsymbol{F}^*\boldsymbol{E}_{xh}^{\mathrm{T}} - \frac{1}{N}\boldsymbol{G}^*\boldsymbol{E}_{xw}^{\mathrm{T}}\right). \tag{6.64}$$

Note that $\boldsymbol{F}^*\boldsymbol{E}_{xh}^{\mathrm{T}}$ is symmetric, the same is true for $\boldsymbol{G}^*\boldsymbol{E}_{xw}^{\mathrm{T}}$ – it follows directly from (6.18) and from the definition of the problem, where the columns of $\boldsymbol{F}^*$ and $\boldsymbol{G}^*$ span different subspaces, hence $\boldsymbol{F}^*\boldsymbol{E}_{xh}^{\mathrm{T}} \neq \boldsymbol{G}^*\boldsymbol{E}_{xw}^{\mathrm{T}}$, thus each of the two terms have to be symmetric. In the light of the previous statement it should be mentioned that $\boldsymbol{\Sigma}_{/F}$ given in (6.59) is not necessarily symmetric since both $\boldsymbol{F}\boldsymbol{E}_{xh}^{\mathrm{T}}$ and $\boldsymbol{E}_{xh} = N\boldsymbol{C}_B\boldsymbol{T}_F^{\mathrm{T}}$ involve the same estimate of $\boldsymbol{F}$ (see the discussion related to (6.19)). However, $\boldsymbol{\Sigma}_{/F}$ converges to a symmetric matrix.

It should be highlighted that now the time to train PLDA *does not depend* on the size of the dataset. The estimation algorithm is given in Alg.4.

Exact Solution By now we have assumed that $\boldsymbol{T}_F$ given in (6.55) is constant even if the number of representations J_i of an individual changes. This is useful when the number of representations of involved individuals differs widely and recomputing $\boldsymbol{T}_F$ would significantly slower the estimation procedure.

Algorithm 4 PLDA estimation algorithm revisited II

Require: I sets $\boldsymbol{\Lambda}_i = \{\boldsymbol{x}_{ij}\}_{i=1}^{J_i}, i = 1, \ldots, I, \sum_{i=1}^{I} J_i = N$ of distinct representations of an individual i; dimensions D_h, D_w of latent variables $\boldsymbol{h}_i, \boldsymbol{w}_{ij}$; initialization (random) matrices $\boldsymbol{F}_{\text{init}}, \boldsymbol{G}_{\text{init}}, \boldsymbol{\Sigma}_{\text{init}}$; number of iterations N_{iter}; small ρ

// Initialize:

$\boldsymbol{F} = \boldsymbol{F}_{\text{init}}, \boldsymbol{G} = \boldsymbol{G}_{\text{init}}, \boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\text{init}}$

$\boldsymbol{C}_X = \frac{1}{N} \sum_{i,j}^{N} \boldsymbol{x}_{ij} \boldsymbol{x}_{ij}^{\mathrm{T}}$

$\boldsymbol{C}_B = \frac{1}{N} \sum_{i=1}^{I} J_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^{\mathrm{T}}, \boldsymbol{\mu}_i = \frac{1}{J_i} \sum_{\boldsymbol{x}_{ij} \in \boldsymbol{\Lambda}_i} \boldsymbol{x}_{ij}$

for $iter = 1$ to N_{iter} **do**

// Expectation step:

$\boldsymbol{\Sigma}_N^{-1} = (\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}$ // identity (C.1) may be used to facilitate the inversion

$\boldsymbol{T}_F = \left(\boldsymbol{F}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathrm{N}}^{-1} \boldsymbol{F} + \rho \boldsymbol{I}\right)^{-1} \boldsymbol{F}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathrm{N}}^{-1}$

$\boldsymbol{T}_G = \left(\boldsymbol{G}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{G} + \boldsymbol{I}\right)^{-1} \boldsymbol{G}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}$

$\boldsymbol{E}_{xh} = N \boldsymbol{C}_B \boldsymbol{T}_F^{\mathrm{T}}$

$\boldsymbol{\Sigma}_{/F} = \boldsymbol{C}_X - \frac{1}{N} (\boldsymbol{F} \boldsymbol{E}_{xh}^{\mathrm{T}})^{\mathrm{T}}$

$\boldsymbol{E}_{xw} = N \boldsymbol{\Sigma}_{/F} \boldsymbol{T}_G^{\mathrm{T}}$

$\boldsymbol{E}_{hh} = \boldsymbol{T}_F \boldsymbol{E}_{xh}$

$\boldsymbol{E}_{hw} = -(\boldsymbol{F} \boldsymbol{E}_{hh} - \boldsymbol{E}_{xh})^{\mathrm{T}} \boldsymbol{T}_G^{\mathrm{T}}$

$\boldsymbol{E}_{ww} = \boldsymbol{T}_G (\boldsymbol{E}_{xw} - \boldsymbol{F} \boldsymbol{E}_{hw})$

$\boldsymbol{B} = [\boldsymbol{F}, \boldsymbol{G}]$

$\boldsymbol{C}_{yy|x} = (\boldsymbol{B}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{B} + \boldsymbol{I})^{-1}$

// Maximization step:

$\boldsymbol{B} = [\boldsymbol{E}_{xh}, \boldsymbol{E}_{xw}] \left(\begin{bmatrix} \boldsymbol{E}_{hh} & \boldsymbol{E}_{hw} \\ \boldsymbol{E}_{hw}^{\mathrm{T}} & \boldsymbol{E}_{ww} \end{bmatrix} + N \boldsymbol{C}_{yy|x} \right)^{-1}$

$\boldsymbol{\Sigma} = \mathrm{diagz}\left(\boldsymbol{C}_X - \frac{1}{N} \boldsymbol{B} [\boldsymbol{E}_{xh}, \boldsymbol{E}_{xw}]^{\mathrm{T}}\right)$

decompose $\boldsymbol{B}$ to $\boldsymbol{F}$ and $\boldsymbol{G}$

end for

return $\boldsymbol{F}$, $\boldsymbol{G}$ and $\boldsymbol{\Sigma}$

In order to avoid the approximations we need to construct $\boldsymbol{T}_F$ for each distinct J_i. Let

$$\boldsymbol{T}_F(J_i) = \left(\boldsymbol{F}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathrm{N}}^{-1} \boldsymbol{F} + \frac{1}{J_i} \boldsymbol{I}\right)^{-1} \boldsymbol{F}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathrm{N}}^{-1}, \tag{6.65}$$

and let Ω_i be the set of indexes of those individuals i who contain the same number of representations J_i (note that sets $\Omega_1, \ldots, \Omega_{N_J}$ are disjoint, and N_J is the number of distinct values of J_i). Then,

$$\boldsymbol{C}_B(J_i) = J_i \sum_{j \in \Omega_i} \boldsymbol{\mu}_j \boldsymbol{\mu}_j^{\mathrm{T}}. \tag{6.66}$$

Thus, $\boldsymbol{C}_B = 1/N \sum_{i=1}^{N_J} \boldsymbol{C}_B(J_i)$. Noticing that $\boldsymbol{T}_F$ and $\boldsymbol{C}_B$ appear only in $\boldsymbol{E}_{xh}$ and $\boldsymbol{E}_{hh}$, the only change consists in replacing (6.57) and (6.60) with

$$\boldsymbol{E}_{xh} = \sum_{i=1}^{N_J} \boldsymbol{C}_B(J_i) \boldsymbol{T}_F(J_i)^{\mathrm{T}}, \tag{6.67}$$

$$\boldsymbol{E}_{hh} = \sum_{i=1}^{N_J} \boldsymbol{T}_F(J_i) \boldsymbol{C}_B(J_i) \boldsymbol{T}_F(J_i)^{\mathrm{T}}, \tag{6.68}$$

respectively. Now, several covariances $\boldsymbol{C}_B(J_i)$ have to be stored in the memory, and several $\boldsymbol{T}_F(J_i)$ have to be computed. However, if the number N_J of distinct values of J_i is small, the increase in the computational costs is negligible. Otherwise, it is useful to fix J_i to some small value, or e.g. cluster values of J_i into a few clusters and fix one J_i for each cluster.

6.2.4 Verification

In the verification phase two hypotheses are tested [8], namely

- hypotheses $\mathcal{H}_s$ that two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ share the same identity,
- hypotheses $\mathcal{H}_d$ that the identity of two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ differs.

The log-likelihood ratio can be written as

$$\begin{aligned}\mathrm{LLR}(\boldsymbol{x}_1,\boldsymbol{x}_2) &= \log\frac{p(\boldsymbol{x}_1,\boldsymbol{x}_2|\mathcal{H}_s)}{p(\boldsymbol{x}_1|\mathcal{H}_d)p(\boldsymbol{x}_2|\mathcal{H}_d)} = \\ &= \log\mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix}\middle|\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{0}\end{bmatrix},\begin{bmatrix}\hat{\boldsymbol{C}}_X & \boldsymbol{C}_F\\ \boldsymbol{C}_F & \hat{\boldsymbol{C}}_X\end{bmatrix}\right) - \log\mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix}\middle|\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{0}\end{bmatrix},\begin{bmatrix}\hat{\boldsymbol{C}}_X & \boldsymbol{0}\\ \boldsymbol{0} & \hat{\boldsymbol{C}}_X\end{bmatrix}\right),\end{aligned} \tag{6.69}$$

where $\hat{\boldsymbol{C}}_X = \boldsymbol{F}\boldsymbol{F}^{\mathrm{T}} + \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}} + \boldsymbol{\Sigma}$ and $\boldsymbol{C}_F = \boldsymbol{F}\boldsymbol{F}^{\mathrm{T}}$, and $\boldsymbol{x}_1, \boldsymbol{x}_2$ were normalized in advance to have zero mean. As proposed in [81], the $\mathrm{LLR}(\boldsymbol{x}_1,\boldsymbol{x}_2)$ can be rearranged so that

$$\mathrm{LLR}(\boldsymbol{x}_1,\boldsymbol{x}_2) = \boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{x}_1 + \boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{x}_2 + 2\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{\Upsilon}\boldsymbol{x}_2 + \mathrm{const}, \tag{6.70}$$

where formulas for the block inverse (6.38) and inverse identities (C.1), (C.2) were used, and

$$\boldsymbol{\Upsilon} = \hat{\boldsymbol{C}}_X^{-1}\boldsymbol{C}_F(\hat{\boldsymbol{C}}_X - \boldsymbol{C}_F\hat{\boldsymbol{C}}_X^{-1}\boldsymbol{C}_F)^{-1}, \tag{6.71}$$

$$\boldsymbol{\Phi} = -\boldsymbol{\Upsilon}\boldsymbol{C}_F\hat{\boldsymbol{C}}_X^{-1}. \tag{6.72}$$

Realizing that the rank of two matrices after multiplication, where one of them has full-rank and the other has rank r, equals r, thus $\mathrm{rank}(\boldsymbol{\Upsilon}) = \mathrm{rank}(\boldsymbol{\Phi}) = D_h$, leads to an efficient estimation of $\mathrm{LLR}(\boldsymbol{x}_1,\boldsymbol{x}_2)$ [81]. If $D_h < D_x$ then $\boldsymbol{\Upsilon}$ can be decomposed to

$$\begin{aligned}\boldsymbol{\Upsilon} &= [\boldsymbol{U}_{D_h}, \boldsymbol{U}_{D_x-D_h}]\mathrm{DIAG}([\lambda_1,\ldots,\lambda_{D_h},0,\ldots,0])[\boldsymbol{U}_{D_h}, \boldsymbol{U}_{D_x-D_h}]^{\mathrm{T}} = \\ &= \boldsymbol{U}_{D_h}\mathrm{DIAG}([\lambda_1,\ldots,\lambda_{D_h}])\boldsymbol{U}_{D_h}^{\mathrm{T}},\end{aligned} \tag{6.73}$$

where the function $\mathrm{DIAG}(\boldsymbol{z})$ creates a diagonal matrix with entries of $\boldsymbol{z}$ on its diagonal, $\boldsymbol{\Lambda} = \mathrm{DIAG}([\lambda_1,\ldots,\lambda_{D_h}])$ is the diagonal matrix formed by nonzero eigenvalues of $\boldsymbol{\Upsilon}$ and $\boldsymbol{U}_{D_h}$ is the matrix of corresponding eigenvectors of $\boldsymbol{\Upsilon}$. Substituting $\tilde{\boldsymbol{x}}_i = \boldsymbol{U}_{D_h}^{\mathrm{T}}\boldsymbol{x}_i$, $\tilde{\boldsymbol{\Phi}} = \boldsymbol{U}_{D_h}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{U}_{D_h}$ we get

$$\mathrm{LLR}(\boldsymbol{x}_1,\boldsymbol{x}_2) = \tilde{\boldsymbol{x}}_1^{\mathrm{T}}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{x}}_1 + \tilde{\boldsymbol{x}}_2^{\mathrm{T}}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{x}}_2 + 2\tilde{\boldsymbol{x}}_1^{\mathrm{T}}\boldsymbol{\Lambda}\tilde{\boldsymbol{x}}_2 + \mathrm{const}. \tag{6.74}$$

Now, for each enrolled vector $\boldsymbol{x}_i$ we can pre-compute $\alpha_i = \tilde{\boldsymbol{x}}_i^{\mathrm{T}}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{x}}_i$ and store $\grave{\boldsymbol{x}}_i = \boldsymbol{\Lambda}^{1/2}\tilde{\boldsymbol{x}}_i$ for future use. Hence, whenever a verification of two vectors is carried out we only have to multiply two low-dimensional vectors ($\dim(\grave{\boldsymbol{x}}_i) = D_f$) and add two numbers to get

$$\mathrm{LLR}(\boldsymbol{x}_1,\boldsymbol{x}_2) = \alpha_1 + \alpha_2 + 2\grave{\boldsymbol{x}}_1^{\mathrm{T}}\grave{\boldsymbol{x}}_2 + \mathrm{const}. \tag{6.75}$$

Note that in this verification scenario we do not care about the form of the decomposition of $\boldsymbol{x}_1$ or $\boldsymbol{x}_2$ (latent variables $\boldsymbol{h}_i$, $\boldsymbol{w}_{ij}$ stay unknown). The question stated is whether two vectors share the same identity given the subspaces generated by $\boldsymbol{F}$ and $\boldsymbol{G}$.

6.3 Joint Factor Analysis (JFA)

JFA follows a very similar reasoning as PLDA, however it was introduced for the task of speaker recognition by Kenny & Dumouchel [82] independently of PLDA. JFA may be seen as a modified PLDA able to handle the concept of GMM supervectors, which were discussed in Chapter 4. Recall that the set of GMM parameters is given as

$$\boldsymbol{\lambda} = \{\omega_m, \boldsymbol{\mu}_m, \boldsymbol{C}_m\}_{m=1}^{M}, \tag{6.76}$$

where M is the number of Gaussians contained in the GMM, ω_m is the weight, $\boldsymbol{\mu}_m$ is the mean and $\boldsymbol{\Sigma}_m$ is the covariance of the m^{th} Gaussian, and $\dim(\boldsymbol{\mu}_m) = D_x$.

Essentially, in JFA the focus is laid on the decomposition of each speaker's mean supervector $\boldsymbol{\psi}_s = [\boldsymbol{\mu}_{s1}^{\mathrm{T}}, \ldots, \boldsymbol{\mu}_{sM}^{\mathrm{T}}]^{\mathrm{T}}$, obtained in the adaptation process of a UBM, to a speaker dependent part $\boldsymbol{\vartheta}_s$ and channel dependent part $\boldsymbol{\varsigma}_s$ so that

$$\boldsymbol{\psi}_{sh} = \boldsymbol{\vartheta}_s + \boldsymbol{\varsigma}_{sh}, \tag{6.77}$$

$$\boldsymbol{\vartheta}_s = \boldsymbol{m}_0 + \boldsymbol{F}\boldsymbol{h}_s + \boldsymbol{D}\boldsymbol{z}_s, \text{ where } \boldsymbol{h}_s, \boldsymbol{z}_s \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{6.78}$$

$$\boldsymbol{\varsigma}_{sh} = \boldsymbol{G}\boldsymbol{w}_{sh} + \boldsymbol{\epsilon}_{sh}, \text{ where } \boldsymbol{w}_{sh} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}),\ \boldsymbol{\epsilon}_{sh} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}), \tag{6.79}$$

where $h = 1, \ldots, H_s$ denotes the session index of a speaker s. Hence, it is assumed that each speaker was recorded several times on different channels (1 recording = 1 session). The term $\boldsymbol{m}_0$ stands for the mean of supervectors $\boldsymbol{\psi}_s$, $\boldsymbol{F}$ is a $D_xM \times D_h$ rectangular matrix with $D_h = \dim(\boldsymbol{h}_s)$, and $\boldsymbol{G}$ is a $D_xM \times D_w$ rectangular matrix with $D_w = \dim(\boldsymbol{w}_{sh})$.

Note the similarity with the formulation of PLDA given in (6.25). The meaning of matrices in (6.77) is the same as in PLDA, columns of $\boldsymbol{F}$ span the between-speaker subspace, columns of $\boldsymbol{G}$ span the within/session speaker subspace, and $\boldsymbol{\epsilon}$ is some residual noise. A slight difference consists in the involvement of the term $\boldsymbol{D}\boldsymbol{z}_s$, where $\boldsymbol{D}$ is assumed to be a diagonal $D_xM \times D_xM$ matrix and $\boldsymbol{z}_s$ is a latent variable of size equal to the size of $\boldsymbol{\psi}_s$. It can be shown [76] that if $\boldsymbol{F} = \boldsymbol{G} = \boldsymbol{0}$ then $\boldsymbol{\psi}_{sh} = \boldsymbol{m}_0 + \boldsymbol{D}\boldsymbol{z}_s + \boldsymbol{\epsilon}_{sh}$ represents the classical MAP adaptation with a relevance factor equal to $\boldsymbol{D}^{-2}\boldsymbol{\Sigma}$ (to have one relevance factor common for all the dimensions of all Gaussians one can take $\tau = \mathrm{tr}(\boldsymbol{D}^{-2}\boldsymbol{\Sigma})/D_xM$). Hence, the term $\boldsymbol{D}\boldsymbol{z}_s$ is used in order to refine the mean supervector $\boldsymbol{\psi}_{sh}$ when the amount of training data is sufficient, and it also helps in situations when the population of training speakers does not suffice to estimate $\boldsymbol{F}$ reliably [83]. In JFA, the columns of $\boldsymbol{F}$ are known as *eigenvoices* and columns of $\boldsymbol{G}$ as *eigenchannels*, they span the speaker and channel subspace, respectively. It can be a little bit misleading since it does not relate to the eigen-decomposition of a matrix, however the terms have adopted with time. Again, it is assumed that $D_h, D_w \ll D_xM$.

Since $\boldsymbol{\psi}_{sh}$ are the mean supervectors of individual speakers a good approximation of $\boldsymbol{m}_0$ is the UBM mean supervector and a good approximation of $\boldsymbol{\Sigma}$ is the $D_xM \times D_xM$ block-diagonal matrix with blocks given by covariances of UBM Gaussians. Loosely speaking, the prior distribution of supervectors $\boldsymbol{\psi}_{sh} \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{\Sigma})$.

In fact, only accumulated statistics (zero, first, and second moments defined in Section 3.1) of given speaker's data related to a UBM are used instead of the adapted supervector $\boldsymbol{\psi}_s$, the adaptation is implicitly present in the model of JFA (see the discussion of the term $\boldsymbol{D}\boldsymbol{z}_s$ above). Assume that each speaker's voice was recorded several times yielding distinct sets of feature vectors $\boldsymbol{o}_t$ one for each session, and let $\{\boldsymbol{N}_{sh}, \boldsymbol{b}_{sh}, \boldsymbol{E}_{sh}\}_{h=1}^{H_s}$ be the set of statistics of all the sessions of a speaker s given a

UBM with parameters $\boldsymbol{\lambda}$ specified in (6.76), where

$$\begin{aligned}
\boldsymbol{N}_{sh} &= \sum_{t\in\Omega_{sh}} \mathrm{DIAG}\left(\left[\gamma_1(t),\ldots,\gamma_m(t),\ldots,\gamma_M(t)\right]^{\mathrm{T}} \otimes \mathbf{1}_{D_x}\right) \text{ of size } D_xM \times D_xM,\\
\boldsymbol{b}_{sh} &= \sum_{t\in\Omega_{sh}} \left[\gamma_1(t)\boldsymbol{o}_t^{\mathrm{T}},\ldots,\gamma_m(t)\boldsymbol{o}_t^{\mathrm{T}},\ldots,\gamma_M(t)\boldsymbol{o}_t^{\mathrm{T}}\right]^{\mathrm{T}} \text{ of size } D_xM \times 1,\\
\boldsymbol{E}_{sh} &= \sum_{t\in\Omega_{sh}} \mathrm{DIAG}\left(\left[\mathrm{diag}\left(\gamma_1(t)\boldsymbol{o}_t\boldsymbol{o}_t^{\mathrm{T}}\right),\ldots,\mathrm{diag}\left(\gamma_M(t)\boldsymbol{o}_t\boldsymbol{o}_t^{\mathrm{T}}\right)\right]\right) \text{ of size } D_xM \times D_xM,
\end{aligned} \tag{6.80}$$

Ω_{sh} represents the index set of feature vectors belonging to the h^{th} session of speaker s, $\mathbf{1}_{D_x}$ is a D_x dimensional vector of ones, $\gamma_m(t)$ defined in (3.4) is the the posterior probability of m^{th} Gaussian given a feature vector $\boldsymbol{o}_t$, and the function $\mathrm{DIAG}(\boldsymbol{z})$ is used to construct a diagonal matrix $\boldsymbol{Z}$ with input vector $\boldsymbol{z}$ on its diagonal. Statistics (6.80) centralized around a given supervector $\boldsymbol{m}_{\mathrm{c}}$ are given as

$$\begin{aligned}
\bar{\boldsymbol{b}}_{i|\boldsymbol{m}_{\mathrm{c}}} &= \boldsymbol{b}_i - \boldsymbol{N}_i\boldsymbol{m}_{\mathrm{c}},\\
\bar{\boldsymbol{E}}_{i|\boldsymbol{m}_{\mathrm{c}}} &= \mathrm{diagz}(\boldsymbol{E}_i - 2\boldsymbol{b}_i\boldsymbol{m}_{\mathrm{c}}^{\mathrm{T}} + \boldsymbol{N}_i\boldsymbol{m}_{\mathrm{c}}\boldsymbol{m}_{\mathrm{c}}^{\mathrm{T}}),
\end{aligned} \tag{6.81}$$

where $\bar{\boldsymbol{E}}_{i|\boldsymbol{m}_{\mathrm{c}}}$ is a diagonal matrix with diagonal specified in (6.81). Finally,

$$\boldsymbol{N}_s = \sum_{h=1}^{H_s} \boldsymbol{N}_{sh},\ \ \boldsymbol{b}_s = \sum_{h=1}^{H_s} \boldsymbol{b}_{sh},\ \ \boldsymbol{E}_s = \sum_{h=1}^{H_s} \boldsymbol{E}_{sh}, \tag{6.82}$$

where we will make a reasonable assumption that channel/session influences are summed out meeting the requirement for H_s to be appropriately high.

The exhaustive derivation of the estimation formulas of JFA can be found in [76]. However, it demands to compute the correlations of all of the latent variables, which become correlated with each other in their posterior distributions [84] (see also (6.40), where correlations between distinct latent variables are given). This may be computationally demanding, therefore the authors in [78, 83, 84, 85] proposed a decoupled estimation of matrices $\boldsymbol{F}$, $\boldsymbol{G}$ and $\boldsymbol{D}$. Three independent FA models are trained in order

$$\boldsymbol{m}_s = \boldsymbol{F}\boldsymbol{h}_s + \boldsymbol{\epsilon},\ \text{input: } \{\bar{\boldsymbol{b}}_{s|\boldsymbol{m}_0}\}_{s=1}^{S} \tag{6.83}$$

$$\boldsymbol{m}_s^D = \boldsymbol{D}\boldsymbol{z}_s + \boldsymbol{\epsilon},\ \text{input: } \{\bar{\boldsymbol{b}}_{s|\boldsymbol{m}_s+\boldsymbol{m}_0}\}_{s=1}^{S} \tag{6.84}$$

$$\boldsymbol{m}_{sh} = \boldsymbol{G}\boldsymbol{w}_{sh} + \boldsymbol{\epsilon},\ \text{input: } \{\{\bar{\boldsymbol{b}}_{sh|\boldsymbol{\vartheta}_s}\}_{h=1}^{H_s}\}_{s=1}^{S} \tag{6.85}$$

$$\text{and } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$$

differing in types of input data. Since (6.83) is the speaker dependent part of (6.77), independent of the channel, it is trained on the set $\{\bar{\boldsymbol{b}}_{s|\boldsymbol{m}_0}\}_{s=1}^{S}$ – the channel effects should be canceled out during the summation (6.82), and all the supervectors $\boldsymbol{b}_s$ are centered according to their mean $\boldsymbol{m}_0$. Since $\boldsymbol{m}_s^D$ is the residual variance in the speaker dependent space it is estimated using the set $\{\bar{\boldsymbol{b}}_{s|\boldsymbol{m}_s+\boldsymbol{m}_0}\}_{s=1}^{S}$ – this time, each supervector $\boldsymbol{b}_s$ is centered according to its already estimated speaker dependent part $\boldsymbol{m}_0+\boldsymbol{m}_s$. At last, using the already estimated speaker dependent part $\boldsymbol{\vartheta}_s = \boldsymbol{m}_0+\boldsymbol{m}_s+\boldsymbol{m}_s^D$ the channel dependent part (6.85) is trained on the set $\{\{\bar{\boldsymbol{b}}_{sh|\boldsymbol{\vartheta}_s}\}_{h=1}^{H_s}\}_{s=1}^{S}$, where $\bar{\boldsymbol{b}}_{sh|\boldsymbol{\vartheta}_s}$ represents the residual part of each supervector $\boldsymbol{m}_s$ reflecting differences of supervectors between sessions of individual speakers. A good initialization of $\boldsymbol{\Sigma}$ is the covariance of UBM. However, after all the matrices $\boldsymbol{F}$, $\boldsymbol{G}$ and $\boldsymbol{D}$ have been estimated it is appropriate to re-estimate also $\boldsymbol{\Sigma}$ on the set $\{\{\bar{\boldsymbol{b}}_{sh|\boldsymbol{\psi}_{sh}}\}_{h=1}^{H_s}\}_{s=1}^{S}$ with $\boldsymbol{\psi}_{sh} = \boldsymbol{\vartheta}_s + \boldsymbol{m}_{sh}$.

6.3.1 Training

Given an input set of S pairs of statistics $\{\boldsymbol{b}_s, \boldsymbol{N}_s\}_{s=1}^S$ related to a given UBM through (6.80), and given a FA model of the form

$$\boldsymbol{m}_s = \boldsymbol{V}\boldsymbol{y}_s + \boldsymbol{\epsilon}, \;\; \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}), \tag{6.86}$$

where $\boldsymbol{\Sigma}$ consists of diagonal blocks formed by covariance matrices of Gaussians in the UBM, the posterior distribution is given as [84]

$$\begin{aligned}
\boldsymbol{y}_s|\boldsymbol{b}_s, \boldsymbol{N}_s &\sim \mathcal{N}\left(E_y[\boldsymbol{y}_s|\boldsymbol{b}_s, \boldsymbol{N}_s], \boldsymbol{C}_{yy|N_s}\right), \\
\boldsymbol{C}_{yy|N_s} &= \left(\boldsymbol{I} + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{N}_s\boldsymbol{V}\right)^{-1} = \left(\boldsymbol{I} + \sum_{m=1}^{M} N_{sm}\boldsymbol{V}_m^{\mathrm{T}}\boldsymbol{\Sigma}_m^{-1}\boldsymbol{V}_m\right)^{-1}, \\
E_y[\boldsymbol{y}_s|\boldsymbol{b}_s, \boldsymbol{N}_s] &= \boldsymbol{C}_{yy|N_s}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{b}_s = \boldsymbol{C}_{yy|N_s}\sum_{m=1}^{M}\boldsymbol{V}_m^{\mathrm{T}}\boldsymbol{\Sigma}_m^{-1}\boldsymbol{b}_{sm},
\end{aligned} \tag{6.87}$$

where $N_{sm}\boldsymbol{I}$, $\boldsymbol{V}_m$, $\boldsymbol{b}_{sm}, \boldsymbol{\Sigma}_m$ are blocks of $\boldsymbol{N}_s$, $\boldsymbol{V}$, $\boldsymbol{b}_s$, $\boldsymbol{\Sigma}$ related to the m^{th} Gaussian of the UBM, respectively. Note that the covariance (6.87) depends on s through $\boldsymbol{N}_s$ (the number of feature vectors aligned to individual Gaussians), thus it changes with each supervector $\boldsymbol{b}_s$. This was the case also for PLDA, see (6.48).

Maximum Likelihood Estimation (MLE) is similar to the estimation described in Section 6.1 with some minor changes. The estimation formula has the form

$$\boldsymbol{V}_m = \left(\sum_{s=1}^{S}\boldsymbol{b}_{sm}E_y[\boldsymbol{y}_s^{\mathrm{T}}|\boldsymbol{b}_s, \boldsymbol{N}_s]\right)\left(\sum_{s=1}^{S}N_{sm}E_y[\boldsymbol{y}_s\boldsymbol{y}_s^{\mathrm{T}}|\boldsymbol{b}_s, \boldsymbol{N}_s]\right)^{-1}, \tag{6.88}$$

where $E_y[\boldsymbol{y}_s\boldsymbol{y}_s^{\mathrm{T}}|\boldsymbol{b}_s, \boldsymbol{N}_s]$ is computed according to (6.14). The difference from FA and (6.17) is that now distinct dimension-blocks of input vectors/supervectors are weighted. Note that if $N_{sm} = N$ would be constant for all s, m we would have

$$\begin{aligned}
E_y[\boldsymbol{y}_s|\boldsymbol{b}_s, N] &= \left(\frac{1}{N}\boldsymbol{I} + \sum_{m=1}^{M}\boldsymbol{V}_m^{\mathrm{T}}\boldsymbol{\Sigma}_m^{-1}\boldsymbol{V}_m\right)^{-1}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\left(\frac{1}{N}\boldsymbol{b}_s\right) \\
\text{and } \boldsymbol{V} &= \left(\sum_{s=1}^{S}\left(\frac{1}{N}\boldsymbol{b}_s\right)E_y[\boldsymbol{y}_s^{\mathrm{T}}|\boldsymbol{b}_s]\right)\left(\sum_{s=1}^{S}E_y[\boldsymbol{y}_s\boldsymbol{y}_s^{\mathrm{T}}|\boldsymbol{b}_s]\right)^{-1},
\end{aligned}$$

thus the estimation of $\boldsymbol{V}$ assuming supervectors on the input would be identical (none distinctions between dimensions) to the estimation of the FA matrix (6.17) with input vectors $1/N\boldsymbol{b}_s$, and as in PLDA, the prior distribution of latent variables would be weighted according to the number of feature vectors N – the higher the N the lower the influence of the prior (see the discussion on (6.48)).

The update formula for the diagonal matrix $\boldsymbol{\Sigma}$ is mostly the same as in the case of FA and (6.20). It has the form

$$\begin{aligned}
\boldsymbol{\Sigma} &= \operatorname{diagz}\left(\boldsymbol{N}^{-1}\sum_{s=1}^{S}\left(\boldsymbol{E}_s - \boldsymbol{V}E_y[\boldsymbol{y}_s|\boldsymbol{b}_s, \boldsymbol{N}_s]\,\boldsymbol{b}_s^{\mathrm{T}}\right)\right), \text{ thus} \\
\boldsymbol{\Sigma}_m &= \operatorname{diagz}\left(\boldsymbol{N}_m^{-1}\sum_{s=1}^{S}\left(\boldsymbol{E}_{sm} - \hat{\boldsymbol{b}}_{sm}\boldsymbol{b}_{sm}^{\mathrm{T}}\right)\right), \;\; \hat{\boldsymbol{b}}_{sm} = \boldsymbol{V}_m E_y[\boldsymbol{y}_s|\boldsymbol{b}_s, \boldsymbol{N}_s].
\end{aligned} \tag{6.89}$$

wher $\boldsymbol{N} = \sum_s \boldsymbol{N}_s$ The discussion on the form of (6.89) is the same as in Section 6.1.

Note that when only the diagonal of the matrix $\boldsymbol{V} = [V_{ij}]$ has to be estimated (this is the case for the square matrix $\boldsymbol{D}$ from (6.84)), where $\boldsymbol{V}$ has to be square, the update formula (6.88) has the form

$$V_{ii} = \frac{a_i}{c_i}, \tag{6.90}$$

where

$$\boldsymbol{a} = \sum_{s=1}^{S} \operatorname{diag}\left(\boldsymbol{b}_{sm} E_y[\boldsymbol{y}_s^{\mathrm{T}} | \boldsymbol{b}_s]\right), \boldsymbol{c} = \sum_{s=1}^{S} \operatorname{diag}\left(N_{sm} E_y[\boldsymbol{y}_s \boldsymbol{y}_s^{\mathrm{T}} | \boldsymbol{b}_s]\right).$$

An efficient MLE algorithm containing some algebraic improvements and approximations was proposed in [86].

Minimum Divergence Estimation introduced in [87] was proposed as an enhancement of MLE. It does corrections in the subspace formed by $\boldsymbol{V}$ once $\boldsymbol{V}$ was estimated reliably accurate. Assume a reliable estimate $\boldsymbol{V}_0$ thn the re-estimation formula for $\boldsymbol{V}$ is given as

$$\boldsymbol{V} = \boldsymbol{V}_0 \boldsymbol{T}_{yy}^{\mathrm{T}}, \tag{6.91}$$

$$\boldsymbol{T}_{yy}^{\mathrm{T}} \boldsymbol{T}_{yy} = \left(\frac{1}{S} \sum_{s=1}^{S} E_y[\boldsymbol{y}_s \boldsymbol{y}_s^{\mathrm{T}} | \boldsymbol{b}_s, \boldsymbol{N}_s] \right) - \boldsymbol{\mu}_y \boldsymbol{\mu}_y^{\mathrm{T}}, \text{ and } \boldsymbol{\mu}_y = \frac{1}{S} \sum_{s=1}^{S} E_y[\boldsymbol{y}_s | \boldsymbol{b}_s, \boldsymbol{N}_s].$$

It is obvious that (6.91) does not allow to leave the subspace formed by columns of $\boldsymbol{V}_0$. It only rotates the eigenvectors and scales the eigenvalues of $\boldsymbol{V}_0 \boldsymbol{V}_0^{\mathrm{T}}$ within this subspace. Such a technique can be helpful in situations when a system already trained on large corpora is used on a smaller data set not seen during the training of $\boldsymbol{V}$. Then, it is useful to adjust at least the orientation of the subspace to meet different operating conditions [88].

6.3.2 New Vector Enrollment

Suppose that the matrices $\boldsymbol{F}$, $\boldsymbol{D}$, $\boldsymbol{G}$ and $\boldsymbol{\Sigma}$ were already estimated on some large data set, and now a new set of feature vectors $\{\boldsymbol{o}_t\}_{t=1}^{T}$ of a speaker s is given. The task is to get the decomposition of the s^{th} speaker's supervector $\boldsymbol{\psi}_s$ in the form (6.77). At first, statistics $\boldsymbol{N}_s$, $\boldsymbol{b}_s$ defined in (6.80) are accumulated utilizing a given UBM. Subsequently, these statistics are centered and projected onto particular subspaces in order

$$\begin{aligned}
E[\boldsymbol{h}_s | \boldsymbol{b}_s, \boldsymbol{N}_s] &= \left(\boldsymbol{F}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{N}_s \boldsymbol{F} + \boldsymbol{I}\right)^{-1} \boldsymbol{F}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{b}_s - \boldsymbol{N}_s \boldsymbol{m}_0) \Rightarrow \boldsymbol{m}_s = \boldsymbol{m}_0 + \boldsymbol{F} E[\boldsymbol{h}_s | \boldsymbol{b}_s, \boldsymbol{N}_s], \\
E[\boldsymbol{z}_s | \boldsymbol{b}_s, \boldsymbol{N}_s, \boldsymbol{m}_s] &= \left(\boldsymbol{D}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{N}_s \boldsymbol{D} + \boldsymbol{I}\right)^{-1} \boldsymbol{D}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{b}_s - \boldsymbol{N}_s \boldsymbol{m}_s) \Rightarrow \boldsymbol{v}_s = \boldsymbol{m}_s + \boldsymbol{D} E[\boldsymbol{z}_s | \boldsymbol{b}_s, \boldsymbol{N}_s], \\
E[\boldsymbol{w}_s | \boldsymbol{b}_s, \boldsymbol{N}_s, \boldsymbol{v}_s] &= \left(\boldsymbol{G}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{N}_s \boldsymbol{G} + \boldsymbol{I}\right)^{-1} \boldsymbol{G}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{b}_s - \boldsymbol{N}_s \boldsymbol{v}_s),
\end{aligned}$$

yielding a decomposition of the s^{th} speaker's supervector in the form

$$\boldsymbol{\psi}_s = \boldsymbol{m}_0 + \boldsymbol{F} E[\boldsymbol{h}_s] + \boldsymbol{D} E[\boldsymbol{z}_s] + \boldsymbol{G} E[\boldsymbol{w}_s], \tag{6.92}$$

where the conditional part was left out to increase readability. In this scenario we assumed that only one recording of speaker s was given. In case when several sessions H_s would be available we would extract $\boldsymbol{b}_{sh}, \boldsymbol{N}_{sh}$ from each session, use $\boldsymbol{b}_s = \sum_{h=1}^{H_s} \boldsymbol{b}_{sh}$ and $\boldsymbol{N}_s = \sum_{h=1}^{H_s} \boldsymbol{N}_{sh}$ when estimating $E[\boldsymbol{h}_s]$ and $E[\boldsymbol{z}_s]$. Thus $\boldsymbol{v}_s = \boldsymbol{m}_0 + \boldsymbol{F} E[\boldsymbol{h}_s] + \boldsymbol{D} E[\boldsymbol{z}_s]$ would be the same for all the sessions, but the channel dependent part $\boldsymbol{\varsigma}_{sh} = \boldsymbol{G} E[\boldsymbol{w}_{sh} | \boldsymbol{b}_{sh}, \boldsymbol{N}_{sh}, \boldsymbol{v}_s]$ would be different for each supervector $\boldsymbol{b}_{sh}$ of each session h.

6.3.3 Verification

Given an unknown set of feature vectors $\boldsymbol{O}_x = \{\boldsymbol{o}_t\}_{t=1}^{T}$ we wish to compute the log-likelihood

$$\log p(\boldsymbol{O}_x|\boldsymbol{y}_{sx}, \boldsymbol{\Omega}, \boldsymbol{\lambda}) \tag{6.93}$$

where $\boldsymbol{y}_{sx} = [\boldsymbol{h}_s^{\mathrm{T}}, \boldsymbol{z}_s^{\mathrm{T}}, \boldsymbol{w}_{sx}^{\mathrm{T}}]^{\mathrm{T}}$ represents parameters of the s^{th} speaker's model given $\boldsymbol{O}_x$, $\boldsymbol{\Omega} = \{\boldsymbol{m}_0, \boldsymbol{F}, \boldsymbol{D}, \boldsymbol{G}, \boldsymbol{\Sigma}\}$ is the set of JFA parameters, and $\boldsymbol{\lambda} = \{\omega_m, \boldsymbol{m}_{0m}, \boldsymbol{\Sigma}_m\}_{m=1}^{M}$ is the set of UBM parameters (see (2.16)). The speaker dependent part $[\boldsymbol{h}_s^{\mathrm{T}}, \boldsymbol{z}_s^{\mathrm{T}}]^{\mathrm{T}}$ of $\boldsymbol{y}_{sx}$ can be determined at the enrollment time of the speaker, however the channel dependent part $\boldsymbol{w}_{sx}$ depends on (and has to be determined according to) the current set of feature vectors $\boldsymbol{O}_x$ (see previous section).

Let $\boldsymbol{\psi}_{sx} = \boldsymbol{m}_0 + \boldsymbol{m}_{sx}$ be a supervector of size $D_x M$, $\boldsymbol{m}_{sx} = \boldsymbol{V}\boldsymbol{y}_{sx}$ and $\boldsymbol{V} = [\boldsymbol{F}, \boldsymbol{D}, \boldsymbol{G}]$. The log-likelihood (6.93) can be computed in the manner of GMM likelihood as

$$\log p(\boldsymbol{O}_x|\boldsymbol{y}_{sx}, \boldsymbol{\Omega}, \boldsymbol{\lambda}) = \sum_{t=1}^{T} \log \sum_{m=1}^{M} \omega_m \frac{1}{(2\pi)^{D_x/2}|\boldsymbol{\Sigma}_m|^{1/2}} \mathrm{e}^{-0.5(\boldsymbol{o}_t - \boldsymbol{\psi}_{sxm})^{\mathrm{T}} \boldsymbol{\Sigma}_m^{-1} (\boldsymbol{o}_t - \boldsymbol{\psi}_{sxm})}, \tag{6.94}$$

where $\boldsymbol{\psi}_{sxm}$ is a block of $\boldsymbol{\psi}_{sx}$ containing dimensions from $(m-1)D_x + 1$ to $(m-1)D_x + M$. However, this method is quite time consuming [89]. The scoring can be greatly simplified aligning each vector $\boldsymbol{o}_t$ to a Gaussian (e.g. by Viterbi). Then, the GMM likelihood simplifies to [76]

$$\begin{aligned}
\log p(\boldsymbol{O}_x|\boldsymbol{y}_{sx}, \boldsymbol{\Omega}, \boldsymbol{\lambda}) &= \sum_{m=1}^{M} N_{xm} \log \frac{1}{(2\pi)^{D_x/2}|\boldsymbol{\Sigma}_m|^{1/2}} - \frac{1}{2} \sum_{m=1}^{M} \sum_{t \in \boldsymbol{\Lambda}_m} (\boldsymbol{o}_t - \boldsymbol{\psi}_{sxm})^{\mathrm{T}} \boldsymbol{\Sigma}_m^{-1} (\boldsymbol{o}_t - \boldsymbol{\psi}_{sxm}) \\
&= \log p(\boldsymbol{O}_x|\boldsymbol{\lambda}) + h(\boldsymbol{O}_x|\boldsymbol{y}_{sx}, \boldsymbol{\Omega}), \qquad (6.95) \\
\log p(\boldsymbol{O}_x|\boldsymbol{\lambda}) &= \sum_{m=1}^{M} N_{xm} \log \frac{1}{(2\pi)^{D_x/2}|\boldsymbol{\Sigma}_m|^{1/2}} - \frac{1}{2} \sum_{m=1}^{M} \sum_{t \in \boldsymbol{\Lambda}_m} (\boldsymbol{o}_t - \boldsymbol{m}_{0m})^{\mathrm{T}} \boldsymbol{\Sigma}_m^{-1} (\boldsymbol{o}_t - \boldsymbol{m}_{0m}), \qquad (6.96)
\end{aligned}$$

$$h(\boldsymbol{O}_x|\boldsymbol{y}_{sx}, \boldsymbol{\Omega}) = \sum_{m=1}^{M} \boldsymbol{m}_{sxm}^{\mathrm{T}} \boldsymbol{\Sigma}_m^{-1} \sum_{t \in \boldsymbol{\Lambda}_m} (\boldsymbol{o}_t - \boldsymbol{m}_{0m}) - \frac{1}{2} \sum_{m=1}^{M} N_{xm} \boldsymbol{m}_{sxm}^{\mathrm{T}} \boldsymbol{\Sigma}_m^{-1} \boldsymbol{m}_{sxm}, \tag{6.97}$$

where N_{xm} is the number of feature vectors from the set $\boldsymbol{O}_x$ aligned to Gaussian m, $\boldsymbol{\Lambda}_m$ is the index set of feature vectors from the set $\boldsymbol{O}_x$ aligned to Gaussian m, and $p(\boldsymbol{O}_x|\boldsymbol{\lambda})$ is the likelihood of $\boldsymbol{O}_x$ in the UBM described by parameters $\boldsymbol{\lambda}$ assuming an alignment of vectors to Gaussians. Denoting

$$\begin{aligned}
\boldsymbol{b}_x^{\mathrm{V}} &= \left[\sum_{t \in \boldsymbol{\Lambda}_1} \boldsymbol{o}_t^{\mathrm{T}}, \ldots, \sum_{t \in \boldsymbol{\Lambda}_m} \boldsymbol{o}_t^{\mathrm{T}}, \ldots, \sum_{t \in \boldsymbol{\Lambda}_M} \boldsymbol{o}_t^{\mathrm{T}} \right]^{\mathrm{T}}, \\
\boldsymbol{E}_x^{\mathrm{V}} &= \mathrm{DIAG} \left(\left[\sum_{t \in \boldsymbol{\Lambda}_1} \mathrm{diag}\left(\boldsymbol{o}_t \boldsymbol{o}_t^{\mathrm{T}}\right), \ldots, \sum_{t \in \boldsymbol{\Lambda}_M} \mathrm{diag}\left(\boldsymbol{o}_t \boldsymbol{o}_t^{\mathrm{T}}\right) \right] \right), \qquad (6.98) \\
\boldsymbol{N}_x &= \mathrm{DIAG}\left([N_{x1}, \ldots, N_{xm}, \ldots, N_{xM}] \otimes \boldsymbol{1}_{D_x}\right),
\end{aligned}$$

and

$$\begin{aligned}
\bar{\boldsymbol{b}}_{x|\boldsymbol{m}_0}^{\mathrm{V}} &= \boldsymbol{b}_x^{\mathrm{V}} - \boldsymbol{N}_x \boldsymbol{m}_0, \\
\bar{\boldsymbol{E}}_{x|\boldsymbol{m}_0}^{\mathrm{V}} &= \mathrm{diagz}\left(\boldsymbol{E}_x^{\mathrm{V}} - 2\boldsymbol{b}_x^{\mathrm{V}} (\boldsymbol{m}_0^{\mathrm{V}})^{\mathrm{T}} + \boldsymbol{N}_x \boldsymbol{m}_0 \boldsymbol{m}_0^{\mathrm{T}}\right), \qquad (6.99)
\end{aligned}$$

(these are the Viterbi alternatives to (6.80), (6.81)), we get

$$\log p(\boldsymbol{O}_x|\boldsymbol{\lambda}) = \sum_{m=1}^{M} N_{xm} \log \frac{1}{(2\pi)^{D_x/2}|\boldsymbol{\Sigma}_m|^{1/2}} - \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{E}}^{\mathrm{V}}_{x|\boldsymbol{m}_0}), \tag{6.100}$$

$$\log p(\boldsymbol{O}_x|\boldsymbol{y}_{sx}, \boldsymbol{\Omega}, \boldsymbol{\lambda}) = \log p(\boldsymbol{O}_x|\boldsymbol{\lambda}) + \boldsymbol{m}_{sx}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{b}}^{\mathrm{V}}_{x|\boldsymbol{m}_0} - \frac{1}{2}\boldsymbol{m}_{sx}^{\mathrm{T}}\boldsymbol{N}_x\boldsymbol{\Sigma}^{-1}\boldsymbol{m}_{sx}. \tag{6.101}$$

Note that in (6.101) only the last two terms on the right-hand side depend on the speaker s.

Since the channel factor $\boldsymbol{w}_{sh}$ for each new recording and each model is unknown, it is useful to integrate it out [83], thus get rid of the extra computations. For convenience let us adjust the notation $p(\boldsymbol{O}_x|\boldsymbol{h}_s, \boldsymbol{z}_s, \boldsymbol{w}_{sx}) = p(\boldsymbol{O}_x|\boldsymbol{y}_{sx}, \boldsymbol{\Omega}, \boldsymbol{\lambda})$. The integration has the form

$$p(\boldsymbol{O}_x|\boldsymbol{h}_s, \boldsymbol{z}_s) = \int p(\boldsymbol{O}_x|\boldsymbol{h}_s, \boldsymbol{z}_s, \boldsymbol{w}_{sx})\mathcal{N}(\boldsymbol{w}_{sx}|\boldsymbol{0}, \boldsymbol{I})\mathrm{d}\boldsymbol{w}_{sx}, \tag{6.102}$$

where $\mathcal{N}(\boldsymbol{w}_{sx}|\boldsymbol{0}, \boldsymbol{I})$ is the standard Gaussian kernel function of $\boldsymbol{w}_{sx}$. The result is given as

$$\begin{aligned}\log p(\boldsymbol{O}_x|\boldsymbol{h}_s, \boldsymbol{z}_s) = \sum_{m=1}^{M} N_{xm} \log \frac{1}{(2\pi)^{D_x/2}|\boldsymbol{\Sigma}_m|^{1/2}} - \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{E}}^{\mathrm{V}}_{x|\boldsymbol{\upsilon}_s}) - \\ - \frac{1}{2}\log|\boldsymbol{C}^{-1}_{ww|x}| + \frac{1}{2}\|\boldsymbol{C}^{1/2}_{ww|x}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{b}}^{\mathrm{V}}_{x|\boldsymbol{\upsilon}_s}\|^2,\end{aligned} \tag{6.103}$$

where $\boldsymbol{\upsilon}_s = \boldsymbol{m}_0 + \boldsymbol{F}E[\boldsymbol{h}_s] + \boldsymbol{D}E[\boldsymbol{z}_s]$ is the s^{th} speaker model (see (6.92)),

$$\boldsymbol{C}_{ww|N_x} = \left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{N}_x\boldsymbol{G} + \boldsymbol{I}\right)^{-1},$$

and $\boldsymbol{C}^{1/2}_{ww|N_x}$ is obtained from the Cholesky decomposition of $\boldsymbol{C}_{ww|N_x}$. Note that one can integrate out all the latent variables, for further details see [76, 83]. Some efficient approximations made in the verification process leading to a less time consuming scoring can be found in [89].

It should be stated that verification scores in real systems are given mostly as Log-Likelihood Ratios (LLRs), thus two hypothesis are tested. Hypothesis that the new recording does belong to a different speaker against the hypothesis that the recording does belong to the given speaker (same as in PLDA, see Section 6.2.4). The former hypothesis uses the parameters of UBM (the decomposition of UBM supervector is $\boldsymbol{\psi}_{UBM} = \boldsymbol{m}_0 + \boldsymbol{G}\boldsymbol{w}_x$, thus $\boldsymbol{F}\boldsymbol{h} + \boldsymbol{D}\boldsymbol{z} = 0$) and the latter hypothesis uses the speaker model described in the previous section to be evaluated. The final LLR is given as

$$\mathrm{LLR}(s, \mathrm{UBM}) = \log p(\boldsymbol{O}_{\boldsymbol{x}}|s) - \log p(\boldsymbol{O}_{\boldsymbol{x}}|\mathrm{UBM}). \tag{6.104}$$

6.4 Identity Vectors (i-vectors)

The question raised by researchers in [5] was whether the channel space obtained by the JFA decomposition is free from any speaker information. Experiments proved that channel factors still contain enough speaker information that even brings improvements when fused. Therefore a combination of speaker and channel space was proposed leading to a *total variability space* containing simultaneously both speaker and channel variabilities. The speaker model has now the form

$$\boldsymbol{\psi}_{sx} = \boldsymbol{m}_0 + \boldsymbol{T}\boldsymbol{w}_{sx} + \boldsymbol{\epsilon}, \;\; \boldsymbol{w}_{sx} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \; \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}), \tag{6.105}$$

where the index x in $\boldsymbol{w}_{sx}$ illustrates that $\boldsymbol{w}_{sx}$ depends also on the channel, $\boldsymbol{w}_{sx}$ is often called the i-vector and its components are called total factors, $\boldsymbol{m}_0$ is the mean vector of $\boldsymbol{\psi}_s$ (can be again taken

from the UBM), $\boldsymbol{T}$ is the total variability space matrix, and $\boldsymbol{\epsilon}$ is some residual noise with diagonal covariance $\boldsymbol{\Sigma}$.

In order to train $\boldsymbol{T}$ the same algorithm as for JFA is used (described in Section 6.3.1), however now instead of labeling the input data according to speakers and their sessions, it is assumed that each recording represents a different speaker [5]. Loosely speaking, for each recording $\boldsymbol{O}_{sx}$ one supervector $\boldsymbol{b}_{sx}$ is extracted along with the matrix $\boldsymbol{N}_{sx}$ (both given in (6.80)), all the supervectors are pooled together and used to train $\boldsymbol{T}$ without any prior information about the labeling of $\boldsymbol{b}_{sx}$.

In the enrolment phase of a new speaker the number of supervectors of each speaker is equal to the number of sessions of that speaker (in JFA only one speaker model was obtained). An i-vector is extracted according to

$$\bar{\boldsymbol{w}}_{sx} = \left(\boldsymbol{T}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{N}_{sx}\boldsymbol{T} + \boldsymbol{I}\right)^{-1}\boldsymbol{T}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{b}_{sx} - \boldsymbol{N}_{sx}\boldsymbol{m}_0), \tag{6.106}$$

where again $\bar{\boldsymbol{w}}_{sx}$ is the MAP estimate of the true latent variable $\boldsymbol{w}_{sx}$. Since the latent variable $\boldsymbol{w}_{sx}$ is assumed to have standard normal distribution the MAP estimate is the mean and the mode of $p(\boldsymbol{w}_{sx}|\boldsymbol{O}_{sx})$.

6.4.1 Verification

Techniques from previous sections may be utilized. The authors in [90, 5] use SVM with cosine kernel

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{x}_2}{\|\boldsymbol{x}_1\|\,\|\boldsymbol{x}_2\|} \tag{6.107}$$

to train the speaker model, and even the cosine kernel alone was used to get the verification score. Some additional channel normalizations (WCCN, LDA, NAP) are often performed in the total variability space (rather than in the supervector space).

In [91, 6] the verification is based on the PLDA model trained in the total variability space, where the labeled i-vectors are utilized.

6.5 Relation of Factor Analysis (FA) and Nuisance Attribute Projection (NAP)

Since NAP, described in Section 5.7, is used to treat the within-class covariance in the manner of channel compensation and the decoupled JFA does the channel compensation too, let us explore the similarities and dissimilarities of both approaches.

In fact the estimation algorithm used in JFA to handle supervectors differs from the standard FA algorithm only in that it puts weights on distinct dimensional blocks of supevectors. Because of this fact and also to make the relation of NAP and FA more evident rather than the JFA algorithm the FA algorithm will be addressed now.

An obvious difference is that FA is represented by a statistical model (one can generate new samples), whereas NAP is given as a transformation matrix minimizing some objective function. However, both can be expressed as a solution of a Least Square (LS) problem. In the following the consequences of the LS formulation are going to be discussed, and it will be shown in what extent do solutions of NAP and FA overlap [92].

Let $\boldsymbol{X}_s$ be a matrix of N_s vectors of speaker s ordered in its columns, let assume that they were normalized to a zero mean in advance, and let $\boldsymbol{X} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_S]$, $N = \sum_{s=1}^{S} N_s$. For simplicity let further assume that $N_c = N_1 = N_2 = \ldots = N_S$. Thus, the overall within covariance matrix

decomposed in NAP (see (5.12)) is given as $\boldsymbol{C}_{\mathrm{W}} = \sum_{s=1}^{S} N_s \sum_{i=1}^{N_s} \boldsymbol{x}_{si}\boldsymbol{x}_{si}^{\mathrm{T}} = N_c \sum_{i=1}^{N} \boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}$. To simplify the computations let us (without loss of generality) drop the scaling term N_c and let us use the overall within covariance matrix $\boldsymbol{C}_{\mathrm{W}} = 1/N \sum_{i=1}^{N} \boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}$. In order to show the NAP objective function in the form of LS let us rewrite (5.12) to

$$
\begin{aligned}
J_{\mathrm{NAP}}(\boldsymbol{F}) = \mathrm{tr}\left(\boldsymbol{P}\boldsymbol{C}_W\right) = \mathrm{tr}\left((\boldsymbol{I} - \boldsymbol{F}_\perp\boldsymbol{F}_\perp^{\mathrm{T}})\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}\right) = \frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}\left(\boldsymbol{x}_i^{\mathrm{T}}(\boldsymbol{I} - \boldsymbol{F}_\perp\boldsymbol{F}_\perp^{\mathrm{T}})\boldsymbol{x}_i\right) = \\
= \frac{1}{N}\sum_{i=1}^{N}\|(\boldsymbol{I} - \boldsymbol{F}_\perp\boldsymbol{F}_\perp^{\mathrm{T}})\boldsymbol{x}_i\|^2 = \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \boldsymbol{F}_\perp\boldsymbol{F}_\perp^{\mathrm{T}}\boldsymbol{x}_i\|^2 = \\
= \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \boldsymbol{F}_\perp\boldsymbol{z}_i\|^2, \text{ and } \boldsymbol{z}_i = \boldsymbol{F}_\perp^{\mathrm{T}}\boldsymbol{x}_i, \qquad (6.108)
\end{aligned}
$$

where $\boldsymbol{P}$ is given in (5.6), the idempotent property $(\boldsymbol{I} - \boldsymbol{F}_\perp\boldsymbol{F}_\perp^{\mathrm{T}})^2 = (\boldsymbol{I} - \boldsymbol{F}_\perp\boldsymbol{F}_\perp^{\mathrm{T}})$ of a projection matrix was used, and recall that the columns of $\boldsymbol{F}_\perp$ are orthonormal ($\boldsymbol{F}_\perp^{\mathrm{T}}\boldsymbol{F}_\perp = \boldsymbol{I}$), thus $\boldsymbol{F}_\perp\boldsymbol{z}_i$ is the orthogonal projection of $\boldsymbol{x}_i$ onto the subspace formed by columns of $\boldsymbol{F}_\perp$. In fact we could solve (5.6) also iteratively iterating between two steps: 1) fix $\boldsymbol{F}_\perp$ and find the coordinates $\boldsymbol{z}_i$ in the column space of $\boldsymbol{F}_\perp$ for each $\boldsymbol{x}_i$, 2) find a new $\boldsymbol{F}_\perp$ that minimizes $\sum_{i=1}^{N}\|\boldsymbol{x}_i - \boldsymbol{F}_\perp\boldsymbol{z}_i\|^2$. Such an iterative procedure does not guarantee the orthogonality of columns of $\boldsymbol{F}_\perp$, however since the objective (5.12) does not depend on the basis of the subspace formed by columns of $\boldsymbol{F}_\perp$ (see the discussion following after (5.6)), we can perform orthogonalization of columns of $\boldsymbol{F}_\perp$ (e.g. by QR decomposition) after each iteration to make the estimation process more robust.

In the case of FA, the objective function (6.21) written in the form of LS have to be minimized. In order to unify the notations let us use $\boldsymbol{F}$ instead of $\boldsymbol{B}$, however be aware that columns of $\boldsymbol{F}$ in NAP are assumed to be orthogonal, whereas none assumptions are made in FA. Thus (6.21) can be rewritten utilizing the new notation as

$$
\frac{2}{N}\,J_{\mathrm{FA}}(\boldsymbol{F}) = \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \boldsymbol{F}\boldsymbol{z}_i\|^2 + \mathrm{tr}(\boldsymbol{F}\boldsymbol{H}\boldsymbol{F}^{\mathrm{T}}), \qquad (6.109)
$$

where

$$
\boldsymbol{z}_i = (\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{F} + \boldsymbol{I})^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_i, \qquad (6.110)
$$

$$
\boldsymbol{H} = (\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{F} + \boldsymbol{I})^{-1}. \qquad (6.111)
$$

We will come out of conclusions made in [80], where it was shown that generative model (6.1) with an isotropic noise covariance $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$, which maximizes the likelihood (6.4) of input data, is given by the eigenvector decomposition of the data covariance matrix. However we will use a different approach to get more insight into the problematic. Assuming isotropic noise we get

$$
\begin{aligned}
\boldsymbol{z}_i &= (\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{F} + \boldsymbol{I})^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_i = \sigma^{-2}(\sigma^{-2}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} + \boldsymbol{I})^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{x}_i = \\
&= (\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{x}_i, \\
\boldsymbol{H} &= (\sigma^{-2}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} + \boldsymbol{I})^{-1} = \sigma^2(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} + \sigma^2\boldsymbol{I})^{-1}. \qquad (6.112)
\end{aligned}
$$

And the criterion (6.109) can be written in the form (see Appendix B)

$$
\frac{2}{N}\,J_{\mathrm{FA}} = \mathrm{tr}(\boldsymbol{C}_W) - \mathrm{tr}(\boldsymbol{K}_1\boldsymbol{C}_F - \boldsymbol{K}_2), \qquad (6.113)
$$

where $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{Q}$ is obtained using Singular Value Decomposition (SVD), $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \boldsymbol{I}$, and since $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$ is positive semi-definite matrix, $\boldsymbol{D} = [d_{ii}]$ is a diagonal matrix with $d_{ii} \geq 0$), $\boldsymbol{F}_\perp = \boldsymbol{F}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{D}^{-1/2}$,

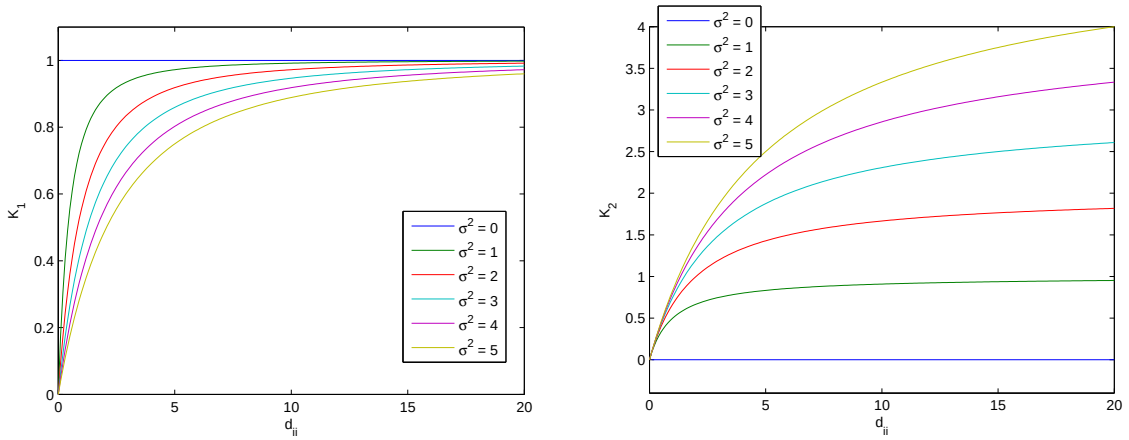

Figure 6.2: The dependency of diagonal entries of $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ on different values of d_{ii} and σ^2.

$\boldsymbol{F}_\perp^{\mathrm{T}}\boldsymbol{F}_\perp = \boldsymbol{I}$, thus $\boldsymbol{F}_\perp\boldsymbol{F}_\perp^{\mathrm{T}}\boldsymbol{x}_i$ is an orthogonal projection onto the subspace spanned by columns of $\boldsymbol{F}$ and $\boldsymbol{F}_\perp^{\mathrm{T}}\boldsymbol{x}_i$ are the coordinates of $\boldsymbol{x}_i$ in this space, $\boldsymbol{C}_F = 1/N\sum_{i=1}^{N}(\boldsymbol{F}_\perp^{\mathrm{T}}\boldsymbol{x}_i)(\boldsymbol{F}_\perp^{\mathrm{T}}\boldsymbol{x}_i)^{\mathrm{T}} = 1/N\,\boldsymbol{F}_\perp^{\mathrm{T}}\boldsymbol{C}_{\mathrm{W}}\boldsymbol{F}_\perp^{\mathrm{T}}$,

$$\boldsymbol{K}_1 = \left[\frac{d_{ii}^2 + 2d_{ii}\sigma^2}{(d_{ii}+\sigma^2)^2}\right],\ \boldsymbol{K}_2 = \left[\frac{d_{ii}\sigma^2}{d_{ii}+\sigma^2}\right] \tag{6.114}$$

are diagonal matrices and $\mathrm{tr}(\boldsymbol{K}_2) = \mathrm{tr}(\boldsymbol{F}\boldsymbol{H}\boldsymbol{F}^{\mathrm{T}})$. Thus, $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ depend on the diagonal matrix $\boldsymbol{D}$, and $\boldsymbol{C}_F$ depends on $\boldsymbol{F}_\perp$. Note that since $\boldsymbol{K}_2$ is a diagonal matrix consisting of singular values of $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$, the second term in (6.109) is responsible only for the scaling of basis vectors of the subspace formed by columns of $\boldsymbol{F}$ according to the level of noise, see Figure 6.2. Examining (6.113) and Figure 6.2 we can make conclusions on the role of $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$. At first, note that the diagonal elements of $\boldsymbol{K}_1$ are lower and upper bounded by 0 and 1, respectively, whereas diagonal elements of $\boldsymbol{K}_2$ are only lower bounded by 0 (recall that $d_{ii} \geq 0$, $\sigma^2 \geq 0$). If $\sigma^2 >> d_{ii}$ then the corresponding directions do not contribute to minimize J_{FA}, and the task of $\boldsymbol{K}_2$ is to completely eliminate these directions.

Since $\boldsymbol{K}_1$, $\boldsymbol{K}_2$ perform only scaling of directions, in order to minimize (6.113) at first $\mathrm{tr}(\boldsymbol{C}_F)$ has to be maximized. This is done when columns of $\boldsymbol{F}_\perp$ are formed by eigenvectors of $\boldsymbol{C}_{\mathrm{W}}$ corresponding to highest eigenvalues, see Section 5.7. A useful side effect is that $\boldsymbol{C}_F$ becomes diagonal with D_y highest eigenvalues λ_i of $\boldsymbol{C}_{\mathrm{W}}$ on its diagonal, where D_y is the latent dimension – number of columns of $\boldsymbol{F}$. To find $\boldsymbol{D}$ (once $\boldsymbol{F}_\perp$ have been found) one has to subsequently maximize $\mathrm{tr}(\boldsymbol{K}_1\boldsymbol{C}_F - \boldsymbol{K}_2)$:

$$\frac{\partial}{\partial d_{ii}}\sum_{i=1}^{D_y}\frac{d_{ii}^2 + 2d_{ii}\sigma^2}{(d_{ii}+\sigma^2)^2}\lambda_i - \frac{d_{ii}\sigma^2}{d_{ii}+\sigma^2} = 0,\ \ i = 1,\ldots,D_y. \tag{6.115}$$

After taking the derivative we get $d_{ii} = 2\lambda_i - \sigma^2$, $\boldsymbol{D}$ is given by eigenvalues of $\boldsymbol{C}_{\mathrm{W}}$. Since $d_{ii} \geq 0$, a condition $d_{ii} = 0$ if $\lambda_i \leq \sigma^2/2$ has to be introduced, which is in accordance with previous discussion on the role of $\boldsymbol{K}_2$.

Before making any judgments on the equivalence of solutions of NAP an FA let express (6.108) as

$$J_{\mathrm{NAP}}(\boldsymbol{F}) = \mathrm{tr}\left((\boldsymbol{I} - \boldsymbol{F}\boldsymbol{F}^{\mathrm{T}})\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}\right) = \mathrm{tr}(\boldsymbol{C}_W) - \mathrm{tr}(\boldsymbol{C}_F). \tag{6.116}$$

Recall that in the case of NAP the solution is also given by the eigenvalue decomposition of $\boldsymbol{C}_{\mathrm{W}}$, thus if the noise model in FA is isotropic the solutions (more precisely the estimated subspaces) for

NAP and FA become identical if for all eigenvalues $\lambda_i > \sigma^2/2$. Otherwise, the FA subspace will be a subspace of the NAP subspace (some directions in FA will have zero gain, i.e. $d_{ii} = 0$). However, criteria J_{NAP} and J_{FA} will still differ in some extent ($\boldsymbol{K}_1$ and $\boldsymbol{K}_2$). If $\sigma^2 = 0$ then $\boldsymbol{K}_1 = \boldsymbol{I}$, $\boldsymbol{K}_2 = \boldsymbol{0}$ and both criteria become equivalent. The same is true if we put an orthonormal restriction on columns of $\boldsymbol{F}$, hence $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{I}\boldsymbol{Q}$ and $d_{ii} = 1$. Now, both $K_1 = (1+2\sigma^2)/(1+\sigma^2)^2$ and $K_2 = \sigma^2/(1+2\sigma^2)$ become constants independent of the choice of $\boldsymbol{F}$, and $J_{\mathrm{FA}} = \alpha_1 J_{\mathrm{NAP}} + \alpha_2$ becomes a scaled version of J_{NAP} for some constants α_1, α_2.

Generally, the FA criterion does incorporate also the influence of noise, thus the value of the criterion differs from J_{NAP} even if the resulting subspaces are identical. To get an idea what is going on when the noise covariance $\boldsymbol{\Sigma}$ is diagonal rather than isotropic we can turn to (6.24). Hence, at first the input data are rescaled according to the given covariance matrix $\boldsymbol{\Sigma}$ (the feature space is adjusted to promote dimensions with low amount of noise), and subsequently the previous discussion can be followed assuming $\sigma^2 = 1$. Note that we have focused on the decomposition of the within covariance matrix $\boldsymbol{C}_{\mathrm{W}}$, thus we were estimating in both cases (NAP and FA) the speaker's session space.

6.6 Conclusion and Remarks

The aim of this chapter was to introduce the concept of Factor Analysis (FA) and present the methods derived from FA. In summary, PLDA is the extended version of FA, where additionally the variance of an individual is treated. The concept of JFA is an alternative to PLDA, but now distinct dimension blocks of given vectors (supervectors) are weighted. Moreover, the between-class variance and within class variance may be estimated independently as discussed in Section 6.3. Also a novel approach to the estimation of PLDA model was proposed that significantly reduces the computational costs. Note that efficient implementations of JFA and i-vector extraction were already studied e.g. in [86, 110, 89].

It should be stated that the discussion in Section 6.5 can be used also when comparing Principal Component Analysis (PCA) and FA. NAP is a special case of PCA, where the input covariance occurring in the objective function is the within covariance computed across several classes of feature vectors.

And finally, note that even if the factor loading matrix $\boldsymbol{B}$ in FA has to be estimated iteratively, if the noise matrix $\boldsymbol{\Sigma}$ is known beforehand or it is isotropic, the columns of $\boldsymbol{B}$ can be computed analytically. If $\boldsymbol{\Sigma}$ is isotropic then $\boldsymbol{B}$ is given in (6.23), and if $\boldsymbol{\Sigma}$ is known beforehand then at first the training data are rescaled as in (6.24) and again $\boldsymbol{B}$ is given in (6.23) with $\sigma^2 = 1$.

Chapter 7

Experiments

In this chapter experiments utilizing techniques from previous chapters will be performed. The focus will be laid on supervectors (SVs) from Chapter 4, more precisely on GMM-mean based, GLDS based and MLLR based SVs. Models for supervectors will be estimated mainly utilizing the Support Vector Machines (SVMs) described in Section 2.3.1. Several distinct kernel types and their influence on the speaker recognition will be analysed along with distinct normalizations of SVs presented in Chapter 5.

Further, system based on i-vectors (see Section 6.4) will be trained and a PLDA model from Section 6.2 will be used as a generative model (hence as a verification tool) in the total variability space. Since the PLDA training procedure proposed in Section 6.2.3 enables a lot of experiments to be performed, the influence of different values of latent dimensions and the influence of distinct development corpora on the PLDA estimation will be analysed.

In fact, many parameters of the system are set empirically by an expert (e.g. number of Gaussian in the GMM, values of latent/reduced dimensions in PLDA/NAP/PCA, etc.) leading to high dimensional SVs. However, it can be anticipated that the true information lies in a much lower dimension. Regrettably, the transformations/functions leading from the extracted feature vector (often containing incomplete information on the subject of interest) to the information itself are highly non-linear and in praxis untraceable. Utilizing only linear transformations the dimensionality reduction can be performed only to some degree (if at all). Methods like PLDA and NAP are inherently reducing the dimension of feature vectors, but also experiments concerning the models based on SVM and the influence of recognition rates on their dimension will be performed and reviewed.

In summary, experiments will be focused on:

1. (baseline) experiments utilizing GMM/UBM based system, where each speaker's GMM is MAP adapted and the Log-Likelihood Ratio (LLR) is computed to get a verification score
2. influence of normalization of SVs on the SVM modelling and speaker recognition
3. dimensionality reduction of SVM models
4. i-vector extraction and PLDA based generative models in the total variability space
5. influence of development speech corpora on the verification rates utilizing PLDA models
6. analysis of the complementarity of mentioned techniques

Results will be presented on two NIST Speaker Recognition Evaluation (SRE) corpora, namely NIST SRE 2008 and NIST SRE 2010 reviewed in the following section. The NIST SRE 2008 will be

Table 7.1: Summary of number of recordings, average number of sessions and number of speakers in distinct corpora.

	female			male		
corpus ID	#recordings	#sessions	#speakers	#recordings	#sessions	#speakers
NIST040506	5500	8	690	3787	8	465
SW1	2311	12	197	2342	11	211
SW2	2862	10	285	2183	10	216
SWC	3753	12	311	2707	12	232
FSH	5774	3	1905	4923	3	1612
overall	20200	-	3388	15942	-	2736

used mostly to calibrate parameters of the speaker recognition system, and the validity of adjusted parameters will be then tested on NIST SRE 2010.

7.1 Used Corpora

In order to be able to perform reliable tests following corpora were utilized: NIST SRE 2004 (NIST04), NIST SRE 2005 (NIST05), NIST SRE 2006 (NIST06), Switchboard 1 Release 2 (SW1), Switchboard 2 Phase 3 (SW2), Switchboard Cellular Audio Part 1 and Part 2 (SWC) and Fisher English Training Speech Part 1 and Part 2 (FSH) for development purposes, and NIST SRE 2008 (NIST08), NIST SRE 2010 (NIST10) were used for calibration and/or testing purposes. Only those speakers from development corpora were used who had more than 4 recorded sessions. In the case of FSH the lower bound on number of sessions was set to 3 (no more sessions were available). Further, the development corpora were divided into following sets:

1. NIST040506 – containing 3787 recordings of 465 males of approximately 8 sessions for each male speaker, and 5500 recordings of 690 females of approximately 8 sessions for each female speaker from NIST04, NIST05 and NIST06

2. SW1 – containing 2342 recordings of 211 males of approximately 11 sessions for each male speaker, and 2311 recordings of 197 females of approximately 12 sessions for each female speaker

3. SW2 – containing 2183 recordings of 216 males of approximately 10 sessions for each male speaker, and 2862 recordings of 285 females of approximately 10 sessions for each female speaker

4. SWC – 2707 recordings of 232 males of approximately 12 sessions for each male speaker, and 3753 recordings of 311 females of approximately 12 sessions for each female speaker

5. FSH – 4923 recordings of 1612 males of approximately 3 sessions for each male speaker, and 5774 recordings of 1905 females of approximately 3 sessions for each female speaker

The summary is given in Table 7.1. Each of the recordings (except those in FSH) had approximately 5 minutes in duration including the silence, length of recordings in FSH was varying from 6 minutes to 12 minutes. The data source of all the recordings was a *telephone conversation*, however in NIST040506 also a few microphone interviews are present. All corpora were recorder in the United States of America. The data are conversational telephone speech in English from all areas of the United States. In the case of SW2 all speakers should be native speakers of English, however this is not true for other corpora. In NIST040506 and NIST08 also other languages than English are present (e.g. Arabic, Russian, Mandarin and Spanish). Speech data for NIST040506, NIST08 and NIST10 were taken mainly from the Mixer Project using the Linguistic Data Consortium's "Fishboard" platform

Table 7.2: Number of speakers and number of trials in NIST SRE 2008 and NIST SRE 2010.

	NIST08		NIST10	
	female	male	female	male
#target speakers	1140	648	1739	1394
#test segment speakers	2498	1535	3267	2474
#trials	28536	16968	103062	74762
#non-target trials	25157	15043	101977	73812
#target trials	3379	1925	1085	950

[93], but also some additional speech data were collected each year specific for each year's evaluation.

In all the tests (for details of the evaluation terminology see Appendix D) "short2-short3" trials from NIST SRE 2008[1] were utilized, and in the case of NIST SRE 2010 "core-core" trials[2] were used. Since development corpora contain only telephone speech (to some minor exceptions) only telephone speech was used in tests (note that in both evaluations also tests utilizing microphone speech are available). The duration of all the test and target recordings in both corpora was approximately 5 minutes including the silence. In all trials only speakers of the same gender were scored against each other. Since the information concerning the gender of the speakers was known, the presented results will be given separately for each gender. Detailed informations on the trials are given in Table 7.2. It should be noted that only one recording for each target and test segment speaker in NIST08, NIST10 was available.

Note that NIST08 will be used mainly for calibration purposes, parameters of individual systems will be tuned mainly on this corpus; NIST10 will be utilized to prove the validity of tuned parameters. Overall more than 4000 hours of speech were processed.

7.2 Feature Extraction

The sample rate of all the telephone speech recordings was 8000 Hz and the sampling format was μ-law. The feature extraction was based on Linear Frequency Cepstral Coefficients (LFCCs), Hamming window of length 25 ms was used, the shift of the window was set to 10 ms. 25 triangular filter banks were spread linearly across the frequency spectrum, 20 LFCCs were extracted, and delta coefficients were added leading to $D = 40$ dimensional feature vectors. Next, Voice Activity Detector (VAD) was used in order to discard the non-speech frames. VAD was based on the detection of energies in filter banks located in the frequency domain. Local Signal-to-Noise Ratios (SNRs) were estimated for each frame as a mean value of SNRs in each of the filter-banks, and global SNR was represented as a mean value of local SNRs computed across whole utterance. Frames with local SNR lower than the global SNR were marked as non-speech. Now, the Feature Warping (FW – see Section 5.3) was applied utilizing a sliding window of length 3 seconds along each dimension. Note that FW was applied after the delta coefficients were added, hence variance in each dimension was 1. To lower the correlation among successive feature vectors all the feature vectors were down-sampled by a factor of 2 at the end.

[1] http://www.itl.nist.gov/iad/mig/tests/spk/2008/sre08_evalplan_release4.pdf
[2] http://www.itl.nist.gov/iad/mig/tests/spk/2010/NIST_SRE10_evalplan.r6.pdf

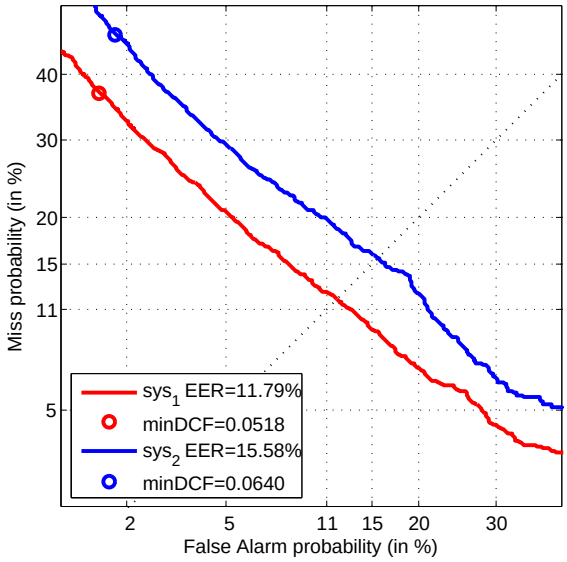

Figure 7.1: Example of DET curves for two different systems, sys_1 performs better than sys_2. Also values of the Equal Error Rate (EER) and the minimum of the Decision Cost Function (minDCF) are given. Note that the values of the threshold θ for which EER and minDCF occurred can not be inferred from the graph.

7.3 Evaluation Methodology

In order to evaluate the performance of the speaker recognition system Detection Error Trade-off (DET) curve [94] will be used. It depicts the dependency of the *miss probability* (target trial is classified as a non-target trial) on the *false alarm probability* (non-target trial is classified as a target trial) for a given verification threshold, moreover error rates plotted on both axes are spread non-linearly so that if the target and non-target scores have Gaussian distribution the DET curve becomes close to linear (i.e. a line) and if both distribution have the same variance the line has a unit slope.

Given a set of trials and a corresponding set of correct results for each trial a point on the DET curve is determined using the following procedure.

1. specify a threshold θ

2. let N_{T} be the number of target trials and let $N_{\text{T}}^{-}(\theta)$ be the number of target trials for which the verification score $< \theta$; the *miss probability* given the threshold θ is equal to $P_{\text{MT}}(\theta) = \frac{N_{\text{T}}^{-}(\theta)}{N_{\text{T}}} \cdot 100[\%]$

3. let N_{NT} be the number of non-target trials and let $N_{\text{NT}}^{+}(\theta)$ be the number of non-target trials for which the verification score $\geq \theta$; the *false alarm probability* given the threshold θ is equal to $P_{\text{FA}}(\theta) = \frac{N_{\text{NT}}^{+}(\theta)}{N_{\text{NT}}} \cdot 100[\%]$

It should be stated that the DET curve does not give any information on the value of the threshold for which a point with coordinates $[P_{\text{FA}}(\theta), P_{\text{MT}}(\theta)]$ was acquired.

In order to express the performance of a system only with one number often the Equal Error Rate (EER) is used, which is the point on the DET curve where $P_{\text{FA}}(\theta)$ and $P_{\text{MT}}(\theta)$ equal.

Another way how to express the error rate with one number specifying some additional prior information is the minimum of the Decision Cost Function (minDCF). DCF is defined as a weighted

sum of miss and false alarm error probabilities[3]:

$$C_{\mathrm{DCF}}(\theta) = C_{\mathrm{MT}} \times P_{\mathrm{MT}}(\theta) \times P_{\mathrm{T}} + C_{\mathrm{FA}} \times P_{\mathrm{FA}}(\theta) \times (1 - P_{\mathrm{T}}), \tag{7.1}$$

where C_{MT} and C_{FA} are the costs of respective detection errors, and P_{T} is the prior probability of the specified target speaker. In this thesis we will use

$$C_{\mathrm{MT}} = 10, \;\; C_{\mathrm{FA}} = 1, \;\; P_{\mathrm{T}} = 0.01 \tag{7.2}$$

used in all the NIST Speaker Recognition Evaluations (SREs)[4]. Note that even if the cost on missing the target speaker is higher than the false alarm, the prior probability of being a target speaker is much smaller. Thus, the minimum of DCF is likely to occur in regions with small values of false alarm. Example of a DET curve is given in Figure 7.1.

7.4 System Setup

UBM At first, two gender dependent Universal Background Models (UBMs) were estimated, one for male and one for female speakers. Each UBM (in fact a GMM) had $M = 1024$ Gaussians and was trained on pooled development corpora NIST040506, SW1, SW2, SWC and FSH from recordings of the respective gender utilizing EM algorithm and 32 reestimations.

GMMs of Speakers GMMs of individual speakers were MAP adapted (see Section 3.2, only means were adapted with a relevance factor $\tau = 14$) using the UBM of respective gender.

Supervector (SV) Extraction Three types of SVs were extracted:

1. $\boldsymbol{\psi}_{\mathrm{GSV}}$ – mean of each speaker's GMM were concatenated into one SV of dimension $M \times D = 1024 \times 40 = 40960$; the mapping is specified in (4.5)

2. $\boldsymbol{\psi}_{\mathrm{GLDS}}$ – monomials up to the $k = 3^{\mathrm{rd}}$ order were expanded, and a $\frac{(D+k)!}{D!k!} = 12341$ dimensional SV was extracted; the mapping is specified in (4.8) and the monomial expansion is given in (4.10)

3. $\boldsymbol{\psi}_{\mathrm{MLLR}}$ – MLLR adaptation of UBM was performed given each speaker's feature vectors, only one (global) MLLR matrix was computed (only one cluster containing all the means was created), the rows of the adaptation matrix $\boldsymbol{W} = [\boldsymbol{A}, \boldsymbol{b}]$ given in (3.13) were concatenated yielding a $(D+1) \times D = 41 \times 40 = 1640$ dimensional SV; the mapping is specified in (4.7)

SVM Note that for each speaker's recording only one SV of each type was extracted. SVM is a binary classifier, thus to train a SVM for each speaker a SVM impostor/background set was constructed (one-against-all training) from all SVs extracted from NIST040506. The main reason was to speed up the estimation process. However, NIST040506 should contain the most similar (and therefore most problematic) speakers to speakers from NIST08 and NIST10 since it is a part of the same corpora (see Section 7.1). In fact, in the training process of a SVM all the vectors lying far from the separating hyperplane are discarded, for more detail see the discussion after (2.33). Since the dimensionality

[3] taken from http://www.itl.nist.gov/iad/mig/tests/spk/2010/NIST_SRE10_evalplan.r6.pdf, but available in each NIST SRE evaluation plan

[4] in NIST SRE 2010 new weighting and prior values were introduced, however the results published in this thesis will not involve them

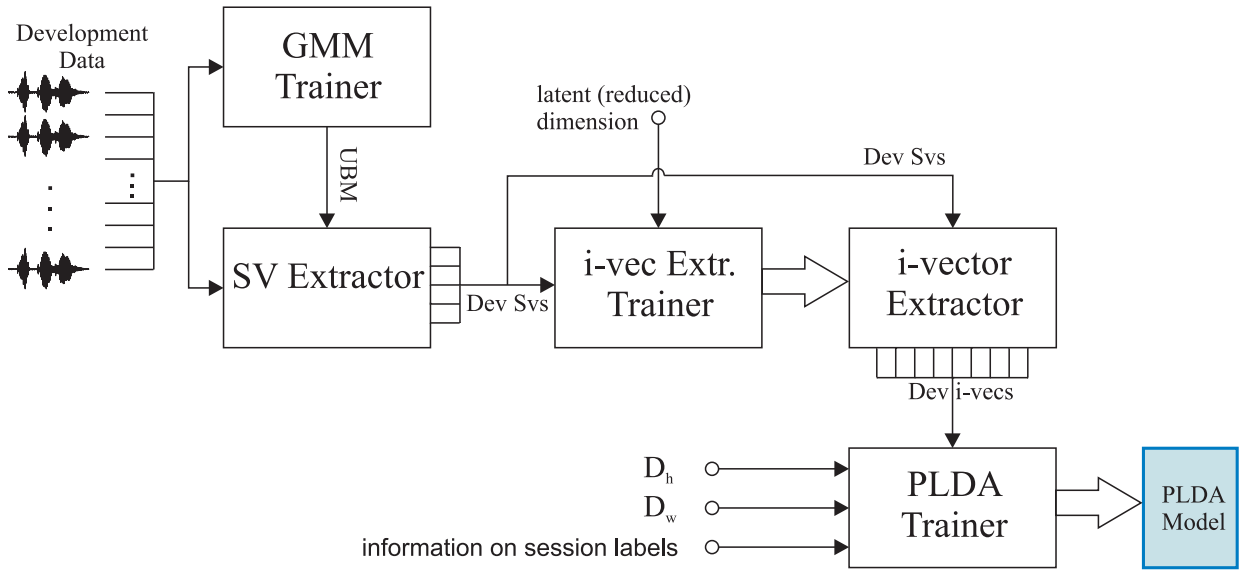

Figure 7.2: Block diagram depicting the estimation procedure of PLDA model. At first, development set containing hundreds of speakers is used to train UBM. Next, Supervectors (SVs) are extracted from the development data, and an i-vector extractor is trained (the dimension of i-vectors have to be specified by the user). Subsequently, i-vectors are extracted from development SVs. Finally, PLDA model is trained from development i-vectors, where the latent dimensions D_h and D_w have to be specified (see Section 6.2) and also information on sessions of speakers have to be given.

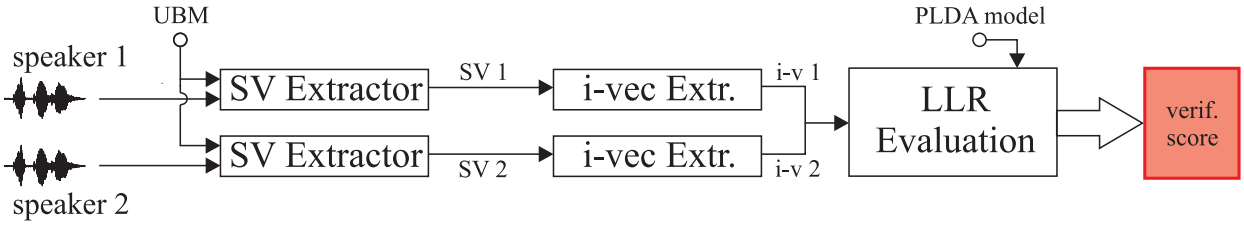

Figure 7.3: Block diagram depicting the verification process. Two utterances of two speakers are given, also a UBM is specified. Two supervectors representing each speaker are extracted and moved to the i-vector extraction phase. At the end, already estimated PLDA model is used to evaluate the Log-Likelihood-Ratio (LLR), defined in Section 6.2.4, of these two i-vectors.

of SVs was high enough to be separated by a hyperplane, only a linear kernel was used. Instead of the kernel trick utilized in the concept of SVM the non-linear mapping was performed explicitly via supervector extraction (for more details see Section 4.2). Thus, the verification consisted only in a scalar multiplication of two vectors – the SV of the hypothesized speaker and the normal vector of the separating hyperplane as given in (2.33). SVM was trained using SVMtorch [20].

NAP In order to train a NAP matrix from Section 5.7 corpora SW1, SW2 and SWC were used because of a lot of available sessions (for some speakers more than 20). Details on the corank of the NAP matrix will be given in Section 7.6. Recall that corank of a matrix is given as $D - D_p$, where $D \times D$ is the size of the projection matrix P and D_p is the rank of this projection matrix, for details see Section 5.7.

Rank Normalization (RN) Experiments with RN of SVs were also performed. SVs were rank normalized along each dimension, and the rank was determined according to a background population of SVs from SW1, SW2 and SWC. For details of RN see Section 5.2.

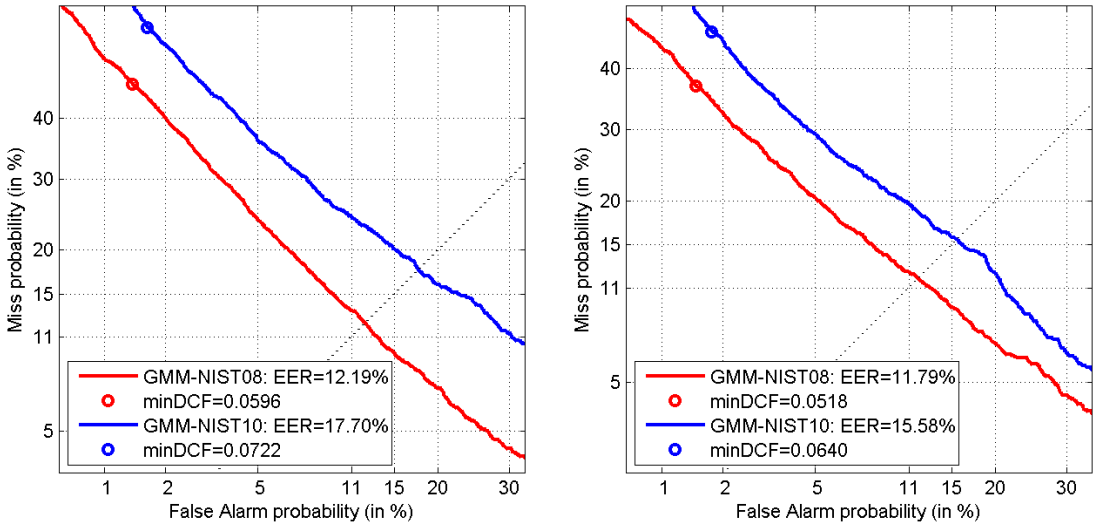

Figure 7.4: DET curves, EERs and minDCF for GMM based system. The left plot depicts performance for females and right plot for males.

i-vectors and PLDA The training procedure of the i-vector extractor and PLDA model in the total variability space is shown in Figure 7.2. Development data to train the UBM were specified above, development data to train the i-vector extractor consisted of corpora NIST040506, SW1, SW2, SWC and FSH (i.e. all the available development data were used). In this case, the block SV Extractor depicted in Figure 7.2 extracts SVs given in (6.80), which are needed to train the total variability space matrix $\boldsymbol{T}$. Seven Maximum Likelihood (ML) iterations and at the end one Minimum Divergence (MD) iteration were carried out, see Section 6.3.1. Subsequently, i-vectors from all development corpora were extracted according to (6.106), normalized to unit lengths [81], and used to train a PLDA model. The PLDA model was trained according to Alg. 4, but rather than fixing the weight ρ of the prior of the latent variable $\boldsymbol{h}_i$, the exact solution described at the end of Section 6.2.3 was used, 50 iterations were performed. Details on values of latent dimensions will be given in upcoming sections. Once a PLDA model was trained the score of a trial was computed as shown in Figure 7.3, where LLR is given in (6.75).

Score Fusion In order to fuse scores of different systems the linear logistic regression from the FoCal tool kit [95] was used. Hence, the combined/fused score was a weighted linear combination of outputs (verification scores) of distinct systems.

7.5 GMM/UBM: Baseline Experiments

GMMs dominated the task of speaker recognition for more than a decade, the concept was introduced by Reynolds in [11]. The verification score of a trial is given as the Log-Likelihood Ratio (LLR)

$$\mathcal{L}_{\text{LLR}} = \sum_{t=1}^{T} \log p(\boldsymbol{x}_t|\boldsymbol{\lambda}_{\text{speaker}}) - \log p(\boldsymbol{x}_t|\boldsymbol{\lambda}_{\text{UBM}}), \tag{7.3}$$

where $\{\boldsymbol{x}_t\}_{t=1}^{T}$ is a set of T feature vectors from the test segment speaker and $\boldsymbol{\lambda}_{\text{speaker}}$, $\boldsymbol{\lambda}_{\text{UBM}}$ are the parameters of the target speaker's and UBM's model, respectively. Verification results are depicted in Figure 7.4.

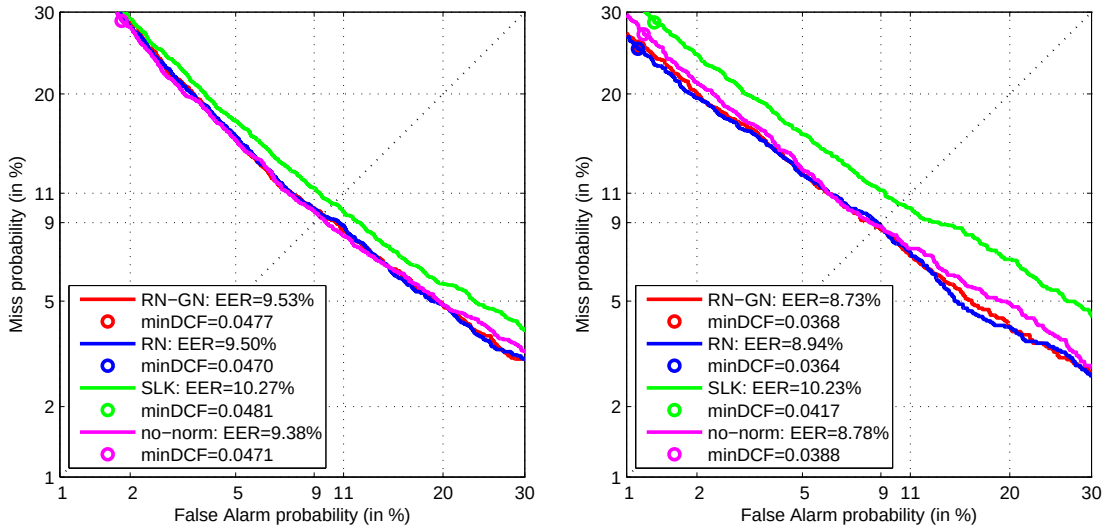

Figure 7.5: DET curves, EERs and minDCF computed on NIST08 for GSV/SVM based system. The left plot depicts performance for females and right plot for males given distinct types of normalizations and kernels.

Nowadays, GMMs play still an important role in the state-of-the-art speaker recognition systems, however they are used mainly to extract data statistics related to distinct parts of the feature space.

7.6 SuperVectors (SVs) and SVM

Experiments will be performed on three types of SVs, namely $\boldsymbol{\psi}_{\text{GSV}}$, $\boldsymbol{\psi}_{\text{MLLR}}$, $\boldsymbol{\psi}_{\text{GLDS}}$, which extraction is described in Section 7.4. Several normalizations will be tested and also the dimensionality reduction will be examined.

7.6.1 Gaussian-mean Supervector (GSV)

Normalizations Since the dimensionality of GSVs is huge, linear kernels are satisfactory and perform best with GSVs [50, 96]. The one often used in the field of speaker recognition is the Supervector Linear Kernel (SLK) introduced in (4.19). In fact it incorporates the whitening step discussed in Section 5.1, but distinct dimension blocks are weighted. Since each dimension block is associated with a Gaussian in the UBM, the weighting relates to the number of feature vectors aligned to this Gaussian. Also Rank Normalization (RN) from Section 5.2 was performed along each dimension of a SV (as proposed in [53]) utilizing a background data set containing SW1, SW2 and SWC, the rank normalized dimensions where then transformed with an inverse normal cumulative distribution function yielding Gauss Normalized (GN) SVs, see Section 5.3. Results on NIST08 can be seen in Figure 7.5. Note that RN-GN is the case of FW when standard normal distribution is used.

Interestingly, best performing system is the one with a simple linear kernel ($K = \boldsymbol{\psi}_{\text{GSV}}^{\text{T}}\boldsymbol{\psi}_{\text{GSV}}$ – in Figure 7.5 denoted as "no-norm"). Hence, no additional normalization helped. This is most probably caused by the FW of feature vectors in the feature extraction phase described in Section 7.2. Since FW is applied independently to each dimension it does retain the correlations between dimensions, moreover also the concentrations of feature vectors in a local area are partially preserved, for several examples of a feature warped dataset see Appendix F. Information on the local variance is captured by the covariance matrix of a Gaussian in the UBM build upon the feature warped feature space. Obviously, the information on the variance in a local area of the feature space is helpful and does not need to be removed. Note that FW (or other variance normalization technique) of feature vectors before the UBM training is of crucial importance when dealing with telephone speech [63].

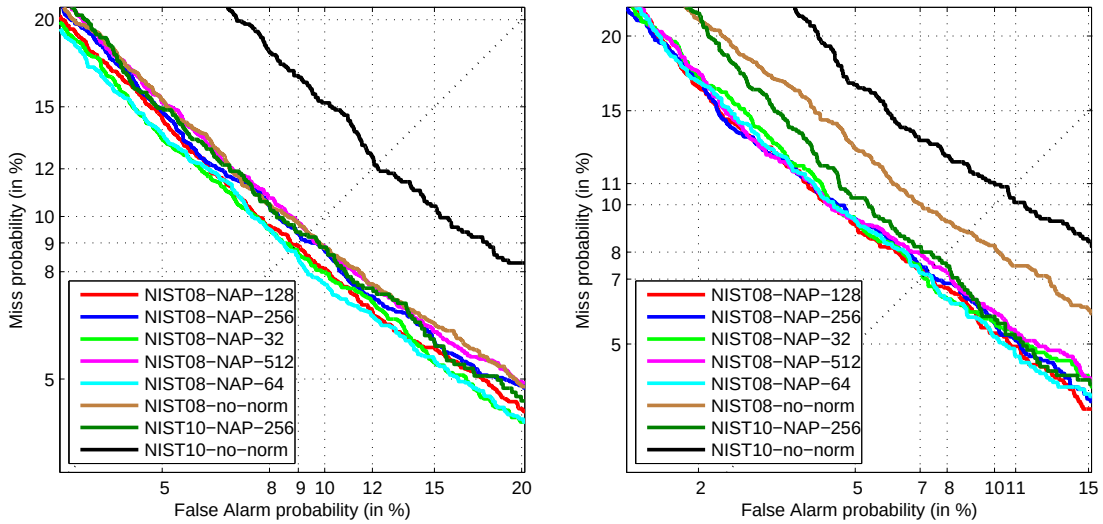

Figure 7.6: DET curves for GSV/SVM based system. The left plot depicts performance for females and right plot for males for different values of corank of the NAP matrix.

Table 7.3: Error rates acquired on female (F) and male (M) parts of NIST08 and NIST10 utilizing the NAP normalization and GSV/SVM based system.

		NIST08						NIST10	
NAP corank		0	32	64	128	256	512	0	256
F	EER [%]	9.38	8.79	8.73	8.94	9.12	9.38	12.17	9.31
	minDCF	0.0471	0.0454	0.0450	0.0455	0.0465	0.0470	0.0548	0.0460
M	EER [%]	8.78	7.06	7.17	7.17	7.27	7.48	10.63	7.68
	minDCF	0.0388	0.0345	0.0345	0.0340	0.0343	0.0346	0.0473	0.0393

NAP Now the influence of the corank of the NAP matrix will be examined. Several values were tested, the results can be found in Figure 7.6 and in Table 7.3 (corank 0 stands for none NAP). The increase in the performance of the speaker recognition system for NIST08 is more evident for male speakers, but error rates decreased for all tested values of the NAP corank. Note that one value of the corank was taken and tested on NIST10. Since the dimensionality of GSV is very high and it is quite hard to predict the behaviour of the system on unseen data the corank 256 was taken (rather than the best performing one) in order to suppress possible undesirable (strong) within-speaker deviations in test data. Note that the decrease of error rates obtained on NIST10 is substantial.

Principal Component Analysis (PCA) We will investigate whether the dimensionality of a SVM model can be further reduced without loosing any information. We could do a dimensionality reduction in the SV space and then train a SVM model for each set of SVs with reduced dimension. However this would be computationally demanding (after each reduction a SVM model has to be estimated) and would not give direct information on the SVM model space. Therefore, a SVM model was trained for each recording from corpora SW1, SW2, SWC, FSH, where the background population of SVs (negative examples) for the training were taken the same as for SVM modelling of NIST08, NIST10 (thus SVs extracted from NIST040506). In order to find a SVM model subspace PCA was performed – eigenvalue decomposition of the covariance matrix computed from SVM models obtained from the development set, eigenvectors corresponding to $D_{\text{red-dim}}$ highest eigenvalues form the columns of the dimensionality reduction matrix $\boldsymbol{F}_\perp$ and $\boldsymbol{F}_\perp^{\mathrm{T}}\boldsymbol{F}_\perp = \boldsymbol{I}$.

Note that SVM model of speaker s is given as $\boldsymbol{\nu}_s = [\boldsymbol{w}_s^{\mathrm{T}}, b_s]^{\mathrm{T}}$, where $\boldsymbol{w}_s$ is the normal of the separating hyperplane and b_s is its offset, both are specified in (2.33). Hence, the covariance matrix $\boldsymbol{C}_\nu$ to be decomposed by PCA is given as

$$\boldsymbol{C}_\nu = 1/S_{\mathrm{d}} \sum_{s\in\boldsymbol{\Omega}_{\mathrm{d}}} (\boldsymbol{\nu}_s - \bar{\boldsymbol{\nu}})(\boldsymbol{\nu}_s - \bar{\boldsymbol{\nu}})^{\mathrm{T}}, \;\; \bar{\boldsymbol{\nu}} = 1/S_{\mathrm{d}} \sum_{s\in\boldsymbol{\Omega}_{\mathrm{d}}} \boldsymbol{\nu}_s, \tag{7.4}$$

where $\boldsymbol{\Omega}_{\mathrm{d}}$ is the set of SVM models trained for recordings from development corpora SW1, SW2, SWC, FSH, S_{d} is cardinality of $\boldsymbol{\Omega}_{\mathrm{d}}$, and $\bar{\boldsymbol{\nu}}$ is the overall mean of SVM models. The SVM model projected onto the subspace formed by columns of $\boldsymbol{F}_\perp$ is given as

$$\boldsymbol{y}_s = \boldsymbol{F}_\perp \boldsymbol{F}_\perp^{\mathrm{T}}(\boldsymbol{\nu}_s - \bar{\boldsymbol{\nu}}) + \bar{\boldsymbol{\nu}} \tag{7.5}$$

(note that $\boldsymbol{y}_s$ and $\boldsymbol{\nu}_s$ are of the same dimension). Since the kernel is linear, given a SV $\boldsymbol{\varrho}_q$ of speaker q extended by one in its last dimension ($\boldsymbol{\varrho}_q \leftarrow [\boldsymbol{\varrho}_q^{\mathrm{T}}, 1]^{\mathrm{T}}$) we get a score

$$score(\hat{\boldsymbol{\nu}}_s, \boldsymbol{\varrho}_q) = \boldsymbol{y}_s^{\mathrm{T}}\boldsymbol{\varrho}_q = (\boldsymbol{\nu}_s - \bar{\boldsymbol{\nu}})^{\mathrm{T}}\boldsymbol{F}_\perp\boldsymbol{F}_\perp^{\mathrm{T}}\boldsymbol{\varrho}_q + \bar{\boldsymbol{\nu}}^{\mathrm{T}}\boldsymbol{\varrho}_q = \hat{\boldsymbol{y}}_s^{\mathrm{T}}\hat{\boldsymbol{\varrho}}_q + \epsilon_q, \tag{7.6}$$

where $\hat{\boldsymbol{y}}_s = \boldsymbol{F}_\perp^{\mathrm{T}}(\boldsymbol{\nu}_s - \bar{\boldsymbol{\nu}})$, $\hat{\boldsymbol{\varrho}}_q = \boldsymbol{F}_\perp^{\mathrm{T}}\boldsymbol{\varrho}_q$ and $\epsilon_q = \bar{\boldsymbol{\nu}}^{\mathrm{T}}\boldsymbol{\varrho}_q$. Hence, both the SVM model and the extended SV are of the same dimension $D_{\text{red-dim}}$, they were both projected to the same SVM model space. The comparison is of course faster than before the projection since the dimensionality of SVs is now lower.

The question is whether the dimensionality reduction does preserve the recognition rates and in what amount. The case for GSV/SVM system is depicted in Figure 7.7 and Table 7.4.

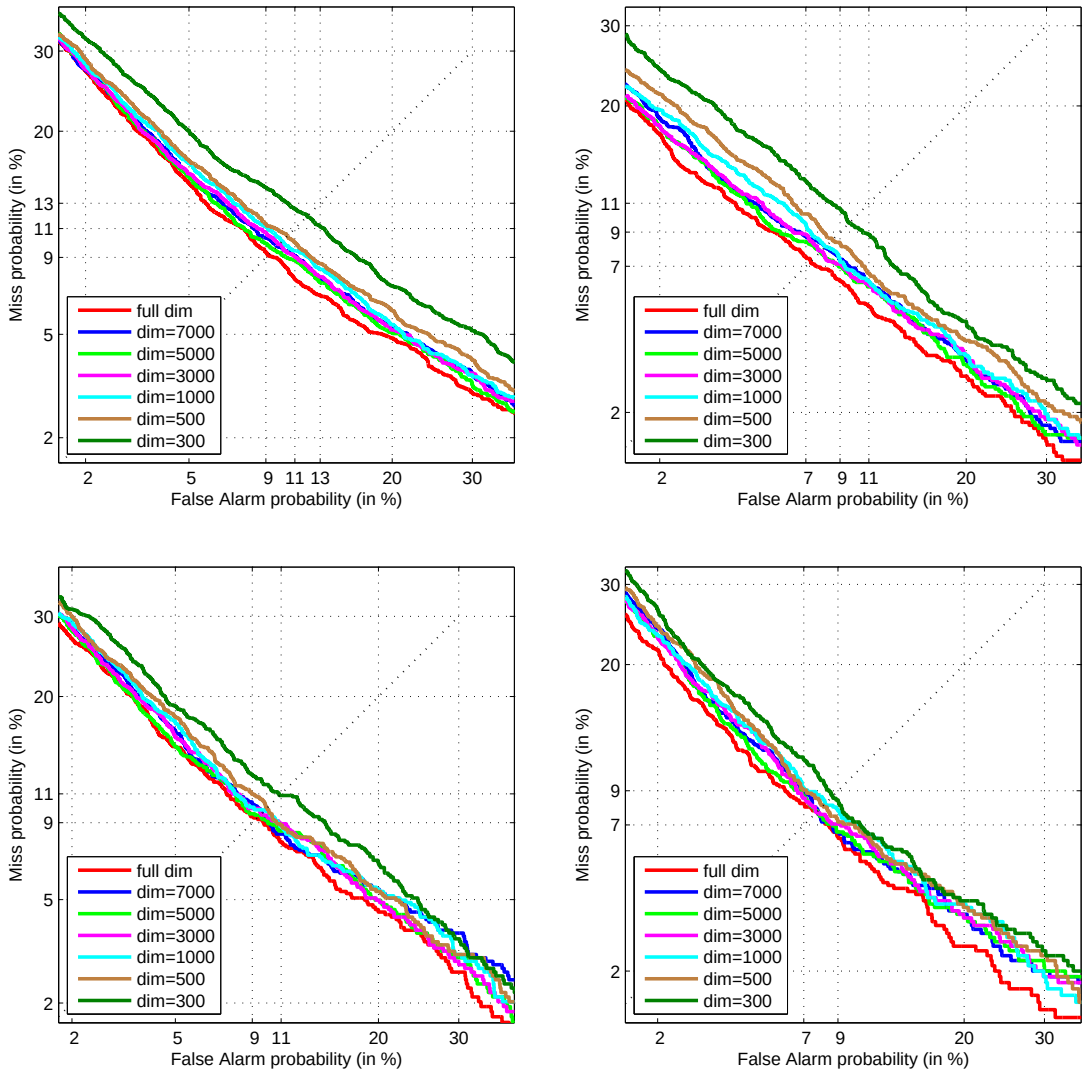

Figure 7.7: DET curves for GSV/SVM based system. In the left column performance for females and in the right column for males is given for different dimensionality reductions of SVM model. 1st row is for NIST08 and 2nd for NIST10.

Table 7.4: Error rates acquired on female (F) and male (M) parts of NIST08 and NIST10 utilizing GSVs and the dimensionality reduction of SVM models. Results are given as EER[%]/(100*minDCF).

	SVM dim	full-dim	7000	5000	3000	1000	500
F	NIST08	9.12/4.65	9.77/4.66	9.53/4.69	9.85/4.65	10.09/4.70	10.48/4.75
	NIST10	9.31/4.60	9.68/4.65	9.49/4.69	9.59/4.65	9.86/4.76	9.95/4.85
M	NIST08	7.27/3.43	8.05/3.61	7.79/3.36	7.84/3.43	8.10/3.56	8.52/3.79
	NIST10	7.68/3.93	7.79/4.30	7.79/4.20	7.68/4.19	8.21/4.17	8.11/4.36

What we did is that we projected the SVM models of target speakers from NIST08 and NIST10 into the models space of development corpora. Error rates increased a little, but the performance is still very high even if the dimension of SVs and SVM model is 8-times lower, in case of NIST10 the system is still performing very well (regarding the "full-dim" system) even for $D_{\text{red-dim}} = 500$. Note that since only approx. 12 000 recordings are available in SW1 + SW2 + SWC + FSH for each

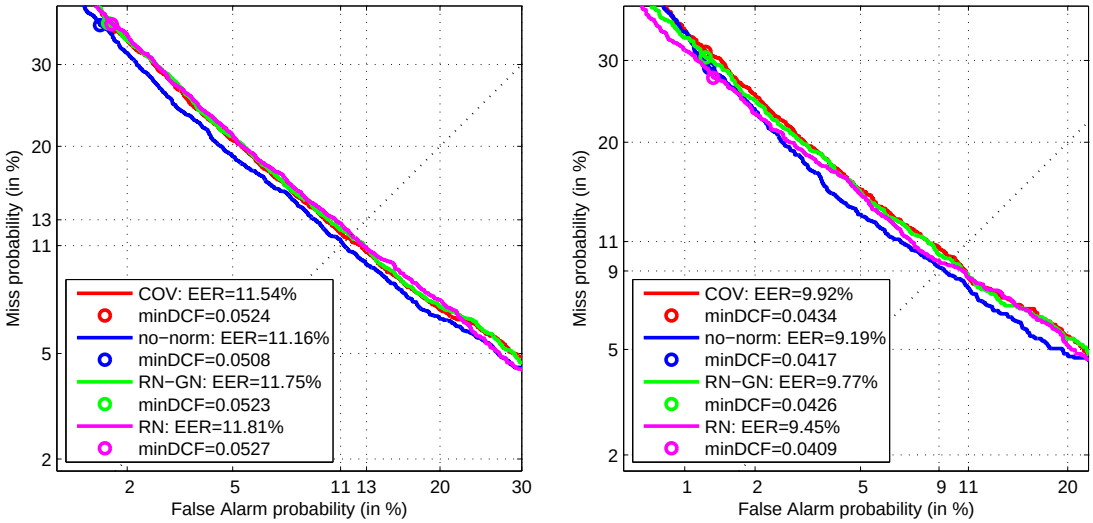

Figure 7.8: DET curves, EERs and minDCF computed on NIST08 for GLDS based system. The left plot depicts performance for females and right plot for males given distinct types of normalizations and kernels.

gender, only 12 000 SVM models were trained. Therefore the highest dimension we could reduce to is around 12 000. It is obvious that the dimensionality of the SVM model space contains a lot of redundant information, and it would be interesting to utilize a SVM kernel that maps into a lower dimensional space instead to higher in cases when such high dimensional SVs are used.

7.6.2 GLDS Supervector

Normalizations In the case of GLDS based SVs at first the same normalizations were tested as in the case of GSV, except the SLK kernel. Instead of the SLK kernel the inverse of the diagonal covariance $\boldsymbol{C}_{\mathrm{SV}}^{-1}$ of SVs computed on development corpora NIST040506, SW1, SW2, SWC was used in the kernel evaluation yielding a kernel function $K = \boldsymbol{\psi}_{\mathrm{GLDS}}^{\mathrm{T}} \boldsymbol{C}_{\mathrm{SV}}^{-1} \boldsymbol{\psi}_{\mathrm{GLDS}}$, for details see (4.2) or Section 5.1. Note that since GLDS is based on a sum of polynomial expansions up to order k of feature vectors $\boldsymbol{o}_t = [o_{t1}, o_{t2}, \ldots, o_{tD}]$ of dimension D (see example (4.10)) that were FW normalized, the first D dimensions of GLDS will be 0 and all the dimensions corresponding to powers $o_{ti}^2, o_{ti}^3, \ldots, o_{ti}^k$ will be constants (after FW vectors $\boldsymbol{o}_t$ have zero mean and unit variance). Hence, the variance computed across the development set will be in these dimensions of GLDS 0 and the inversion $\boldsymbol{C}_{\mathrm{SV}}^{-1}$ will fail. Therefore, all these dimensions were left out when working with GLDS supervectors (they do not contain any information).

Results can be found in Figure 7.8. Again, the best performing system is the one denoted as "no-norm" using a simple linear kernel function $K = \boldsymbol{\psi}_{\mathrm{GLDS}}^{\mathrm{T}} \boldsymbol{\psi}_{\mathrm{GLDS}}$ without any normalizations performed on SVs. The same reasoning as in the previous section can be used.

NAP Also performance of the Speaker Recognition (SR) system in dependence on the value of the corank of the NAP matrix was examined on NIST08, and the corank 64 was taken to carry out experiments on NIST10. Results are given in Figure 7.9 and Table 7.5, 0 stands for no NAP.

PCA Results on the dimensionality reduction of the SVM model are given in Figure 7.10 and Table 7.6. Again, the dimensionality of the SVM model space can be significantly compressed without greater loss in the performance.

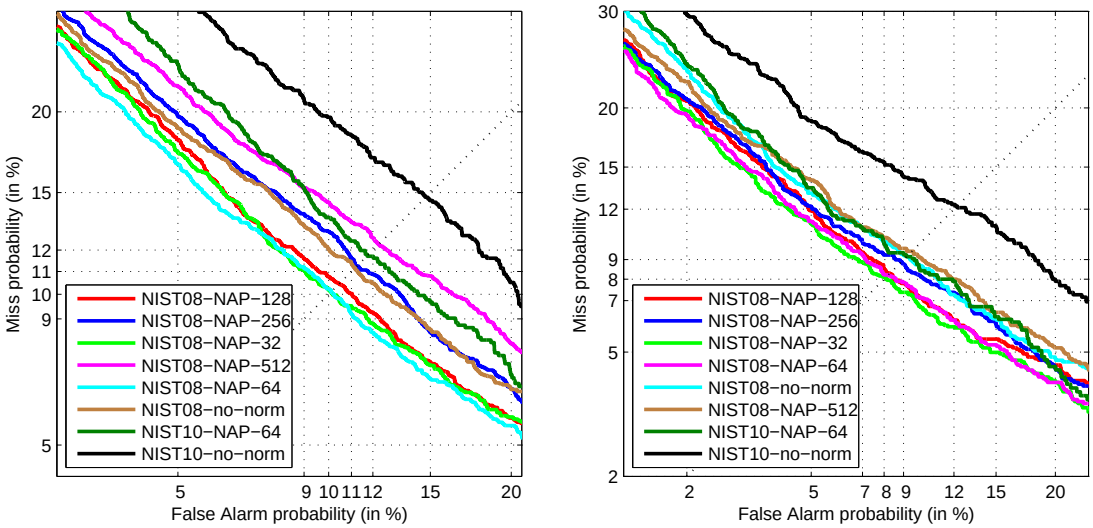

Figure 7.9: DET curves for GLDS/SVM based system. The left plot depicts performance for females and right plot for males for different values of corank of the NAP matrix.

Table 7.5: Error rates acquired on female (F) and male (M) parts of NIST08 and NIST10 utilizing the NAP normalization and GLDS/SVM based system.

		NIST08						NIST10	
NAP corank		0	32	64	128	256	512	0	64
F	EER [%]	11.16	10.09	10.12	10.45	11.25	12.28	14.75	11.71
	minDCF	0.0508	0.0485	0.0481	0.0486	0.0492	0.0509	0.0605	0.0569
M	EER [%]	9.19	8.05	8.21	8.26	8.88	9.45	12.21	9.16
	minDCF	0.0417	0.0376	0.0365	0.0379	0.0375	0.0385	0.0485	0.0430

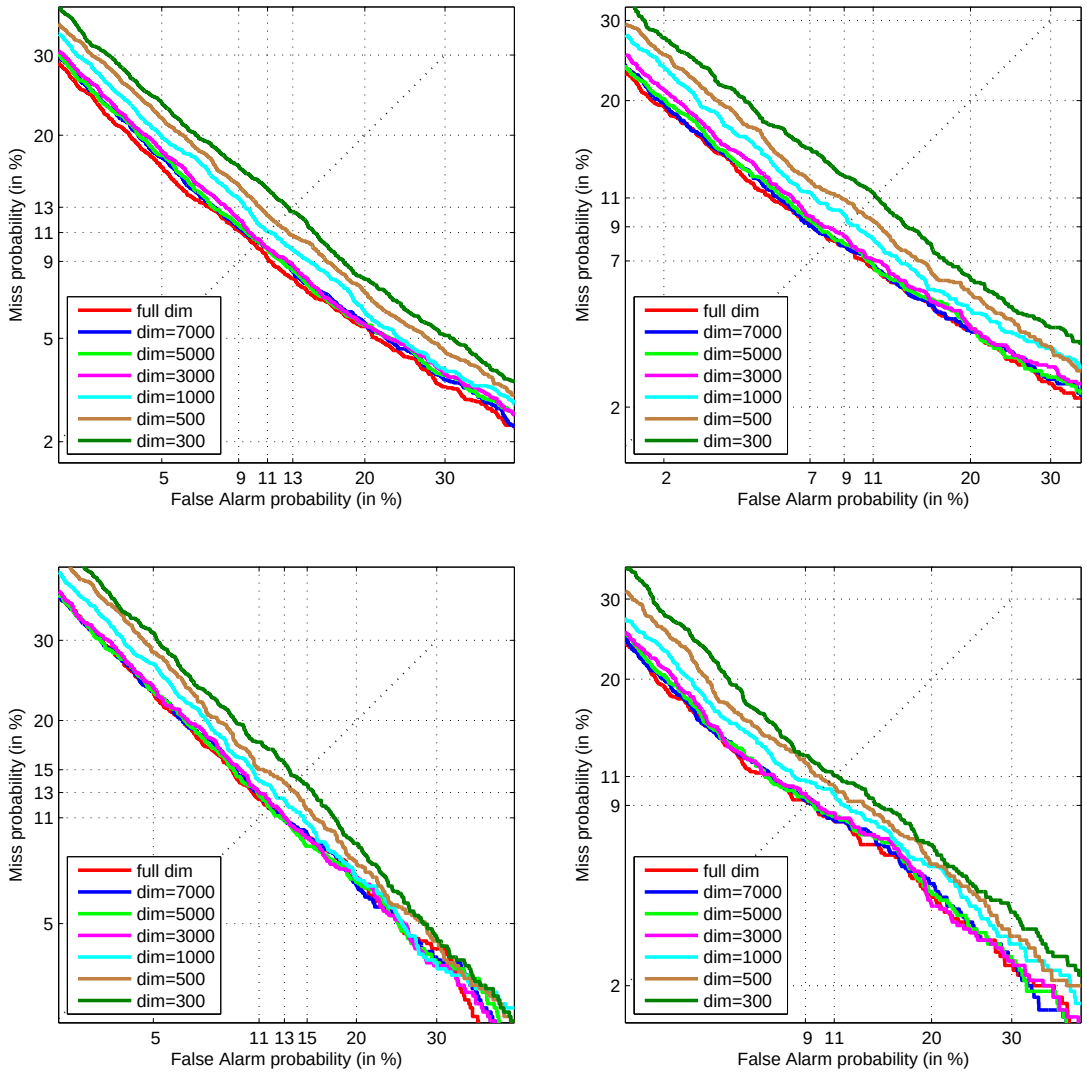

Figure 7.10: DET curves for GLDS/SVM based system. In the left column performance for females and in the right column for males is given for different dimensionality reductions of SVM model. 1st row is for NIST08 and 2nd for NIST10.

Table 7.6: Error rates acquired on female (F) and male (M) parts of NIST08 and NIST10 utilizing GLDS SVs and the dimensionality reduction of SVM models. Results are given as EER[%]/(100*minDCF).

	SVM dim	full-dim	7000	5000	3000	1000	500
F	NIST08	10.12/4.81	10.33/4.90	10.30/4.92	10.42/4.95	11.07/5.15	11.75/5.34
	NIST10	11.71/5.69	11.89/5.73	11.80/5.74	11.98/5.82	12.53/6.02	13.46/6.26
M	NIST08	8.21/3.65	8.31/3.74	8.31/3.74	8.62/3.89	9.25/4.11	10.03/4.22
	NIST10	9.16/4.30	9.26/4.33	9.26/4.40	9.37/4.42	10.11/4.69	10.63/5.02

7.6.3 MLLR Supervector

Normalizations The one-class alternative described in Section 4.3.2 was utilized. The same normalizations as in previous sections were tested, namely: Rank Normalization (RN), Gauss Normalization applied after RN (RN-GN), the One-Class Kernel (OCK) given in (4.34) utilizing a block-diagonal

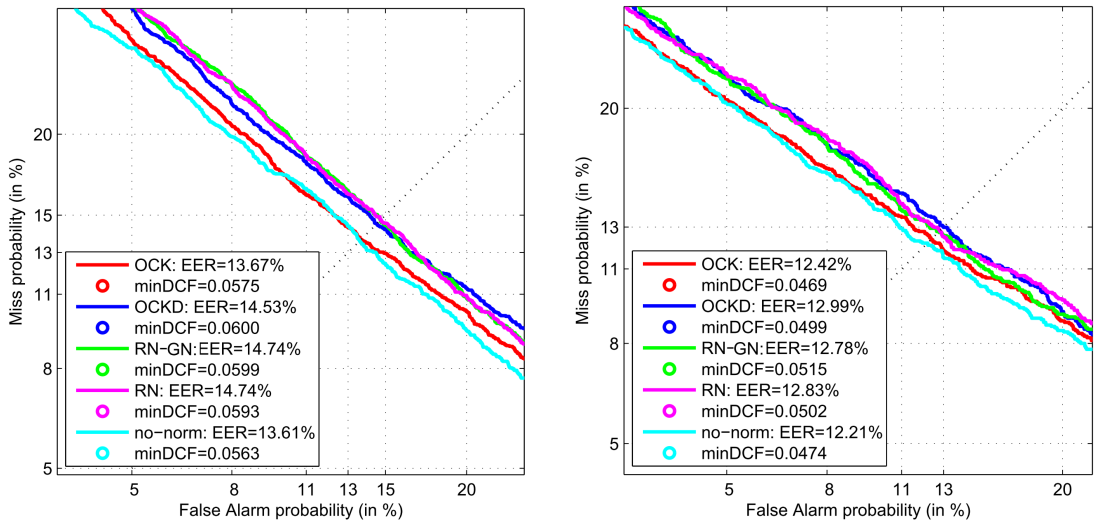

Figure 7.11: DET curves, EERs and minDCF computed on NIST08 for MLLR/SVM based system. The left plot depicts performance for females and right plot for males given distinct types of normalizations and kernels.

normalization matrix; OCKD is the OCK alternative, where only the diagonal from the block-diagonal matrix was taken. Results are given in Figure 7.11.

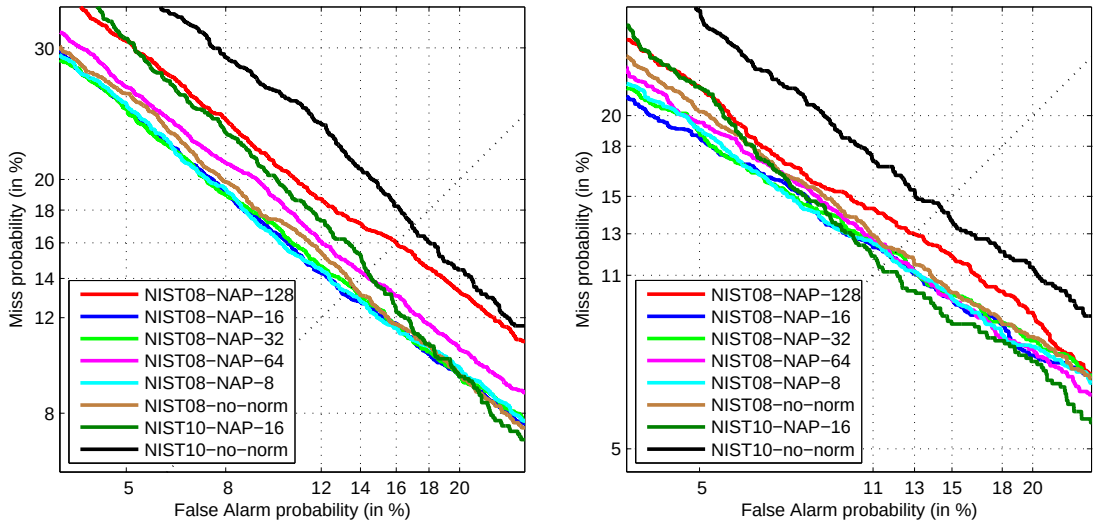

Figure 7.12: DET curves for MLLR/SVM based system. The left plot depicts performance for females and right plot for males for different values of corank of the NAP matrix.

Table 7.7: Error rates acquired on female (F) and male (M) parts of NIST08 and NIST10 utilizing the NAP normalization and MLLR/SVM based system.

		NIST08						NIST10	
NAP corank		0	8	16	32	64	128	0	16
F	EER [%]	13.61	13.29	13.17	13.55	14.23	16.01	16.96	14.38
	minDCF	0.0563	0.0553	0.0552	0.0569	0.0597	0.0617	0.0696	0.0652
M	EER [%]	12.21	11.84	11.90	12.05	11.84	12.99	14.42	11.37
	minDCF	0.0474	0.0444	0.0442	0.0455	0.0459	0.0497	0.0607	0.0547

The best performing system is again the one utilizing a simple kernel function $K = \boldsymbol{\psi}_{\text{MLLR}}^{\text{T}} \boldsymbol{\psi}_{\text{MLLR}}$ without any further normalization of SVs.

NAP Tests with the corank of the NAP matrix are given in Figure 7.12 and Table 7.7. Note that the contribution to the recognition on NIST08 is not apparent (however it does not spoil the performance), but it does significantly improve the error rates of recognition on NIST10.

PCA Results are given in Figure 7.13 and Table 7.8. Even if the dimensions of MLLR SVs are very low in comparison to GSVs or GLDS SVs, the dimension can be reduced to its half without any loss.

The contribution of MLLR SVs to the recognition is lower than the contribution of GSVs and GLDS SVs, the causes are: the low dimensionality of MLLR SVs, the fact that only one MLLR transformation matrix was used, and that the UBM was adapted instead of adaptation of an complex HMM based system. Much lower error rates can be acquired when LVCSR system is in use [97, 53].

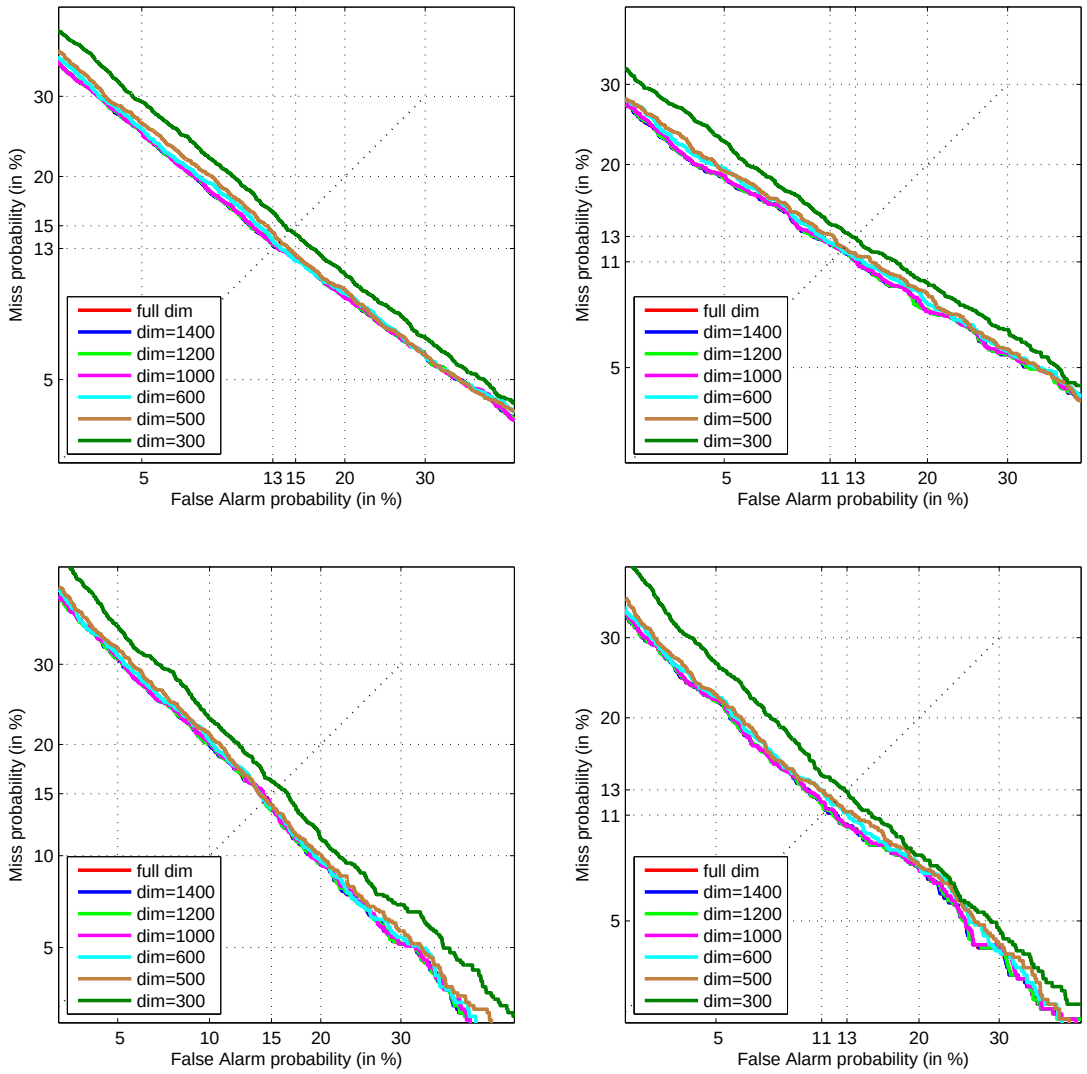

Figure 7.13: DET curves for MLLR/SVM based system. In the left column performance for females and in the right column for males is given for different dimensionality reductions of SVM model. 1st row is for NIST08 and 2nd for NIST10.

Table 7.8: Error rates acquired on female (F) and male (M) parts of NIST08 and NIST10 utilizing MLLR SVs and the dimensionality reduction of SVM models. Results are given as EER[%]/(100*minDCF).

	SVM dim	full-dim	1400	1200	1000	600	500
F	NIST08	13.17/5.52	13.17/5.52	13.20/5.53	13.23/5.53	13.32/5.59	13.61/5.75
	NIST10	14.38/6.52	14.38/6.53	14.38/6.52	14.65/6.52	14.38/6.65	14.47/6.73
M	NIST08	11.90/4.42	11.84/4.41	11.84/4.41	11.90/4.42	11.90/4.50	12.10/4.58
	NIST10	11.37/5.47	11.37/5.48	11.37/5.47	11.47/5.51	11.79/5.56	12.11/5.67

7.7 PLDA and i-vectors

The crucial problem when proposing a speaker verification system composed of modules (e.g. JFA, PLDA) is that data from a lot of speakers are required, moreover several sessions have to be available for each speaker in order to train a reliable i-vector extractor and a PLDA model. The problem faced in this section will address the question whether distinct speech corpora (SW1, SW2, NIST040506, SWC) should be pooled together and used to train one PLDA model, or if each corpus should be used individually to train a separate PLDA model. In the latter scenario the results are fused at the end.

In fact, PLDA (along with all the other FA based techniques and NAP) does the decomposition of a covariance matrix. Hence, the question can be reformulated whether the overall covariance of all the development data should be used, or if the covariance of one corpus should be handled separately. If each covariance is handled separately (one PLDA model is acquired for each development corpus) the contribution of the decomposition of each corpora's covariance to the recognition is handled in the fusion phase. Loosely speaking, if the PLDA model leads to poor recognition rates, the fusion weights used to weight this system's score get lower.

Since often the verification conditions are unknown in advance (e.g. in the Speaker Recognition Evaluations (SREs) organized by NIST and other institutions) we cannot count on the use of one specific speech corpus performing best on the development set. It is more convenient to utilize several corpora, and in a way described in the previous paragraph handle inappropriate deviations in acoustic conditions of distinct corpora.

We will now analyse the performance of PLDA models in dependence on different amounts of development data. Experiments are an extension of results published in [98]. Moreover, the dependence on the dimension D_h and D_w of latent variables $\boldsymbol{h}_i$ and $\boldsymbol{w}_{ij}$ will be examined. In order to decrease number of graphs to be depicted only EERs will be given. At first the behaviour of the system utilizing corpora NIST040506, SW1, SW2 will be discussed, and subsequently corpora SWC will be added to the development data.

7.7.1 Development Corpora: NIST040506, SW1, SW2

At first all the development data are pooled together and one PLDA model is trained. Next, one PLDA model is trained for each corpus (hence, three models are trained in total), each system based on one PLDA model is then scored against trials in NIST08. Given true identities of each pair of speakers scored in the trials the logistic linear regression is used to estimate the Fusion Coefficients (FCs) related to each PLDA model. Finally, to prove the validity of learned FCs trials from NIST10 are scored, results are depicted in Figure 7.14.

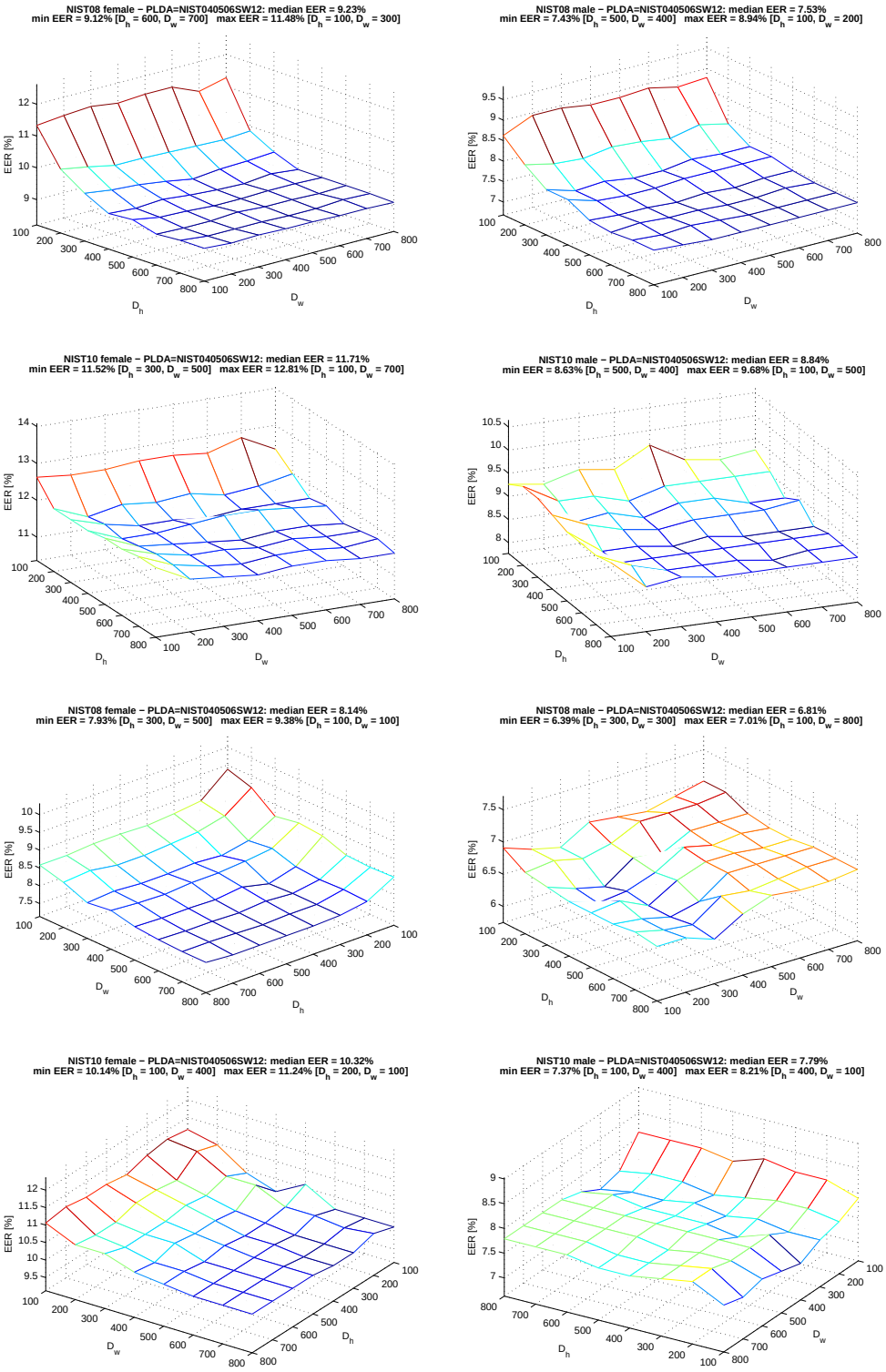

Figure 7.14: *First four plots*: One PLDA model trained on pooled corpora NIST040506, SW1 and SW2. Also the maximal, minimal and median value of EER are given along with dimension of latent variables for which they occur (given in brackets). Results were computed on NIST08 (1^{st} row) and NIST10 (2^{nd} row) for females (1^{st} column) and males (2^{nd} column). *Second four plots*: Three distinct PLDA models were trained, one for each corpora: NIST040506, SW1 and SW2. Subsequently the verification scores for trials from NIST08 were evaluated and the linear logistic regression was utilized in order to compute the fusion coefficients. These coefficient were then used to fuse scores obtained on NIST08 (3^{rd} row) and NIST10 (4^{th} row)

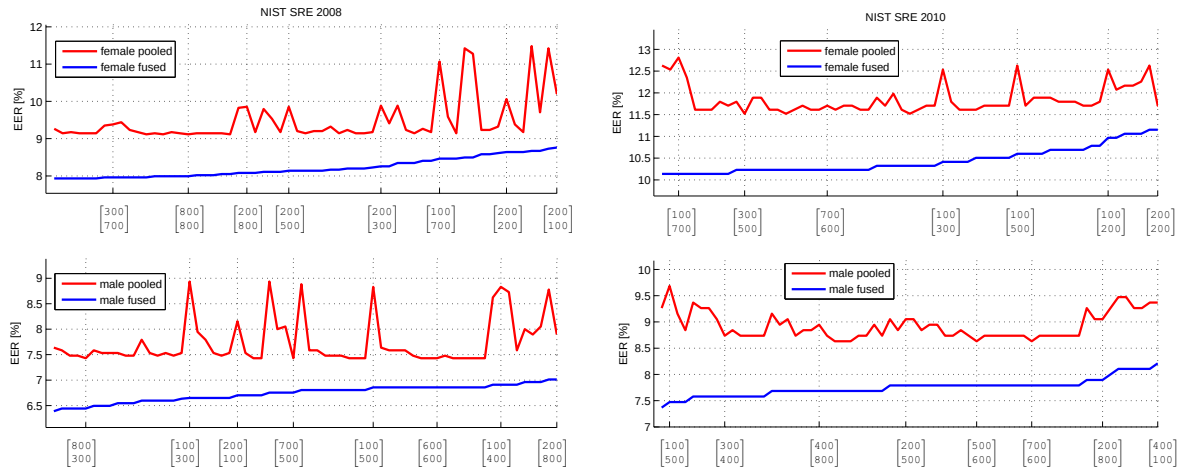

Figure 7.15: Plots depict the comparison of situations when development corpora are pooled and fused. In the left half EERs obtained on NIST08 are compared, and in the right half EERs obtained on NIST10 are compared. In order to plot the graph EERs from the fused system were sorted in ascending order (blue line). The red line represents EERs from the pooled system that correspond to the values of dimensions of latent variables D_h and D_w for which the value of EER on the blue line occurred. Note that the vectors on the x-axis are couples $[D_h, D_w]^{\mathrm{T}}$, but since the lines are sorted according the values of EER of the fused system, they are in any order.

The question is which of the systems, the pooled or the fused one, performs better. One could compare graphs in Figure 7.14, however a visual inspection of these graphs is quite difficult. Therefore Figure 7.15 was created, where EERs of the fused system (blue line) were sorted, and for each value of EER and dimension D_h and D_w for which this value occurred, value of EER of the pooled system (red line) in point $[D_h, D_w]^{\mathrm{T}}$ is plotted. For several values of EER also the dimensions D_h and D_w of latent variables are specified on the x-axis. Again, results are depicted for NIST10 as well as NIST08. Note that the fused system performs better in each of the conditions (for each dimension D_h and D_w), moreover variations in error rates related to distinct latent dimensions are lower.

7.7.2 Development Corpora: NIST040506, SW1, SW2, SWC

Now the SWC corpus is added, and the previous experiments will be repeated.

Results related to the pooled corpora and to the fused system (fusion coefficients trained again on NIST08) can be found in Figure 7.16, and a easy-to-inspect comparison of fused and pooled systems is depicted in Figure 7.17. Note that the fused system does still outperform the pooled one in the case of female speakers, but for male speakers this is no longer true for a substantial amount of values of latent dimensions. However, the fused system does not get worse than the pooled one, and for certain values of D_h and D_w the fused system still performs best. Apparently, having a larger development set leads to more efficient cancelling/averaging of undesirable acoustic deviations in this set.

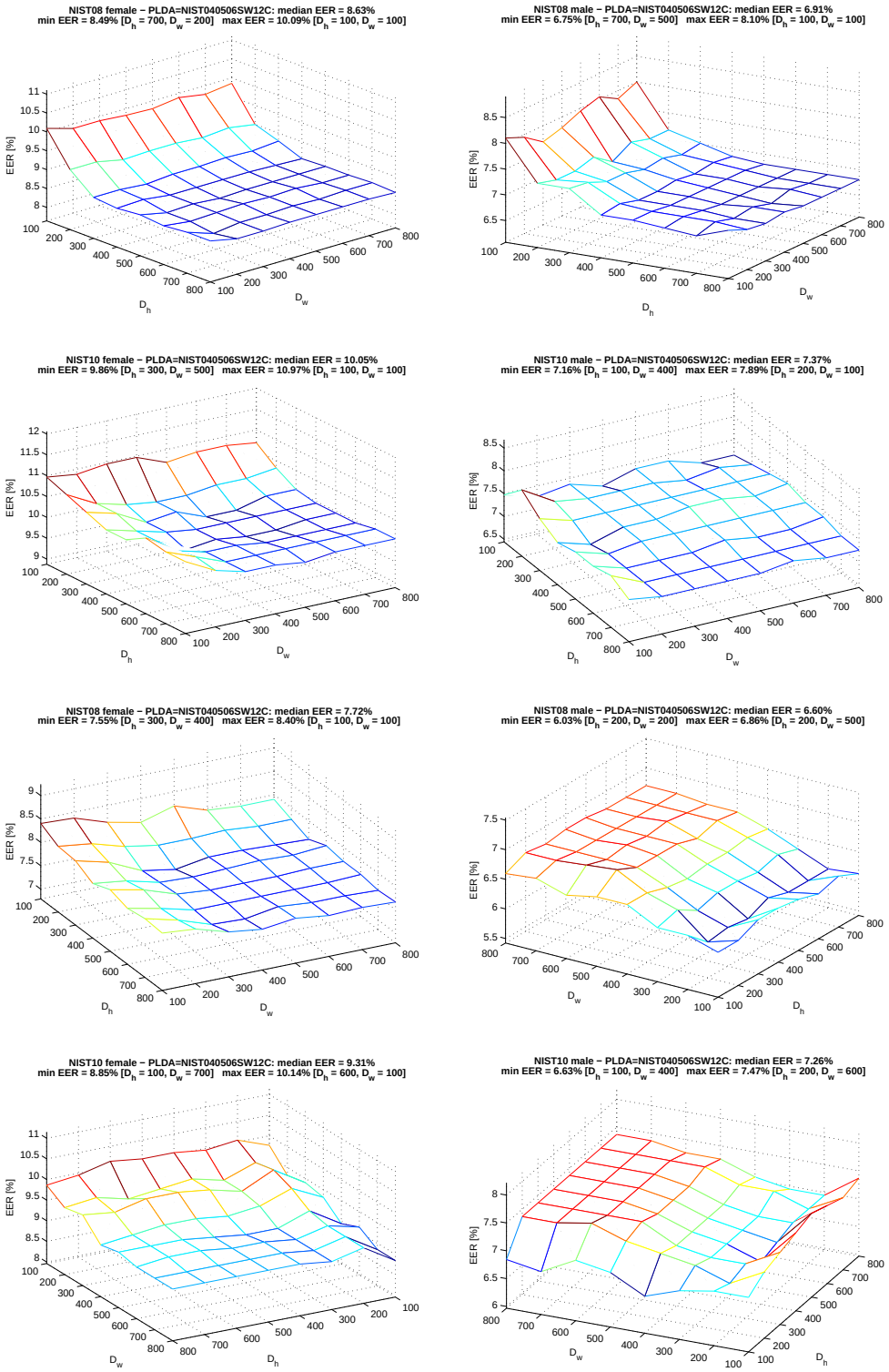

Figure 7.16: *First four plots*: One PLDA model trained on pooled corpora NIST040506, SW1, SW2 and SWC. Also the maximal, minimal and median value of EER are given along with dimension of latent variables for which they occur (given in brackets). Results were computed on NIST08 (1^{st} row) and NIST10 (2^{nd} row). *Second four plots*: Four distinct PLDA models were trained, one for each corpora: NIST040506, SW1, SW2 and SWC. Subsequently the verification scores for trials from NIST08 were evaluated and the linear logistic regression was utilized in order to compute the fusion coefficients. These coefficient were then used to fuse scores obtained on NIST08 (3^{rd} row) and NIST10 (4^{th} row).

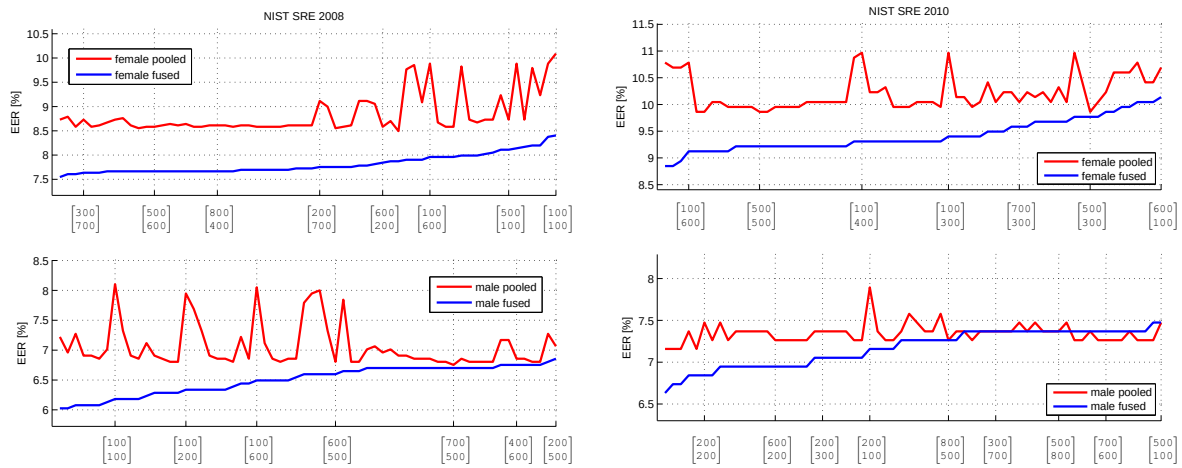

Figure 7.17: Plots depict the comparison of situations when development corpora are pooled and fused. In the left half EERs obtained on NIST08 are compared, and in the right half EERs obtained on NIST10 are compared. For details of the plot see the text or caption of Figure 7.15.

Table 7.9: Time durations in seconds needed to process 10 iterations of a PLDA training algorithm.

algorithm	naive	$T_{F,G}$	exact	nearly-exact
t [s]	4820.23	46.48	12.82	1.92

7.7.3 Summary

After inspection of figures from previous section and additional figures given in Appendix E we can infer that more stable/robust regions (in the sense of value of EER) are for female speakers related to higher values of latent dimension of D_h and D_w, in the case of male speakers it is more convenient to choose D_w small. Hence, mainly in cases with lower amount of training data it is preferable to adhere to such values of D_h and D_w.

And finally, let us mention the time durations needed to process 10 iterations of the PLDA estimation algorithm. Time given in seconds needed to process 8312 i-vectors from male speakers from corpora NIST040506 + SW1 + SW2 (one i-vector for one recording, where the number of recordings in individual corpora can be found in Table 7.1) is given in Table 7.9. All the implementations are described in Section 6.2. The naive implementation follows the implementation from [8], where a huge matrix has to be reassembled for each distinct number of session of a speaker, and is given in Alg. 2. Note that for each distinct number of sessions the matrix was inverted only once and stored for the future use. Method denoted $T_{F,G}$ follows Alg. 3, where the inversion of the huge matrix was replaced by an inversion of two substantially smaller matrices and $\rho = 1/8$ was fixed to a small value to avoid ill-conditioning. The method "nearly-exact" completely follows Alg. 4, where at first covariances (6.53) and (6.54) are estimated and the PLDA estimation does approach the data only through these covariances, moreover $\rho = 1/8$ is fixed. The exact case follows Alg. 4, but ρ is not fixed, instead a covariance matrix (6.66) is accumulated and stored for each distinct number of sessions and the projection matrix (6.65) is used.

Algorithms were implemented in MATLAB® R2011b (7.13.0.564), computer with Intel Core2 Quad CPU 2.83GHz and 8GB RAM was used, and only one thread was run.

Note that in all the experiments on PLDA the exact case was used (thus no approximations where made) and 50 re-estimations of PLDA model were performed. Since the speed burst of the algorithm proposed in this thesis is huge it was possible to carry out high amount of experiments (more than

4000 PLDA models were trained processing 50 iterations for each model).

7.8 Complementarity Analysis

In order to investigate the complementarity of SVM based systems and i-vector based system, outputs (verification scores) of these systems will be fused. For this purpose the logistic linear regression from the FoCal tool kit [95] will be utilized. Hence, the fused score will be given as a linear combination of scores obtained from individual systems. To train the Fusion Coefficients (FCs) data and trials from NIST08 will be utilized, and the learned FCs will be then used to fuse outputs of systems trained for NIST10. Let us summarize the main ideas and dissimilarities of methods examined in this thesis:

1. i-vectors combined with a PLDA model are used to find a low dimensional representation of a Supervector (SV) similar to GMM-mean SV (GSV), moreover PLDA decomposes the feature space into speaker- and session-dependent parts

2. i-vectors and PLDA model are generative and do not discriminate between speakers, whereas SVM as a discriminative classifier does; note that even if PLDA is a discriminative model it discriminates between the speaker- and the channel-subspace

3. presented SVs used with SVM incorporate different kinds of information

4. GSV is build from a set of vectors pointing to positions in the feature space with increased concentration of feature vectors; these vectors are concatenated to a high dimensional SV

5. in the case of GLDS the covariance and higher order moments of the whole speaker's data set are extracted; as stated in Section 7.6.2, since FW is applied on low dimensional vectors in the features space in the phase of feature extraction (see Section 7.2), the first D dimensions in all GLDS SVs are zero (no information) – truly only the covariance and higher order moments contribute to the recognition

6. MLLR is used to transform all the means of a UBM in order to fit given feature vectors, therefore information contained in the MLLR SVs can be thought of as a "model error" (UBM error) given a (spekaer's) feature set

Therefore the complementarity of presented methods should be preserved. In the fusion systems GSV-NAP-256, GLDS-NAP-64, MLLR-NAP-16 (trained via SVM with a simple linear kernel) performing best in experiments from previous sections will participate along with the fused i-vec/PLDA system trained on NIST040506, SW1, SW2 and SWC, which result are depicted in Figure 7.16. Inspecting Figure 7.16 and the results of i-vec/PLDA system we can notice that the setting $D_h = 100$, $D_w = 600$ does give good results in all four cases (male/female and NIST08/NIST10). Therefore the i-vec/PLDA system used in the fusion will be based upon these values of latent dimensions; recall that D_h is the dimension of the between-identity subspace and D_w is the dimension of the within-identity subspace.

Results are shown in Figure 7.18 for NIST08, for NIST10 in Figure 7.19, and the error rates are summarized in Table 7.10. The fusion of systems is undoubtedly beneficial since error rates decreased in all cases. However, the performance of the MLLR based system is too poor in comparison with the other systems and the fusion does not contribute any further to the recognition performance. In fact, this is also the case with the baseline GMM system from Section 7.5, which was because of clarity omitted from the results.

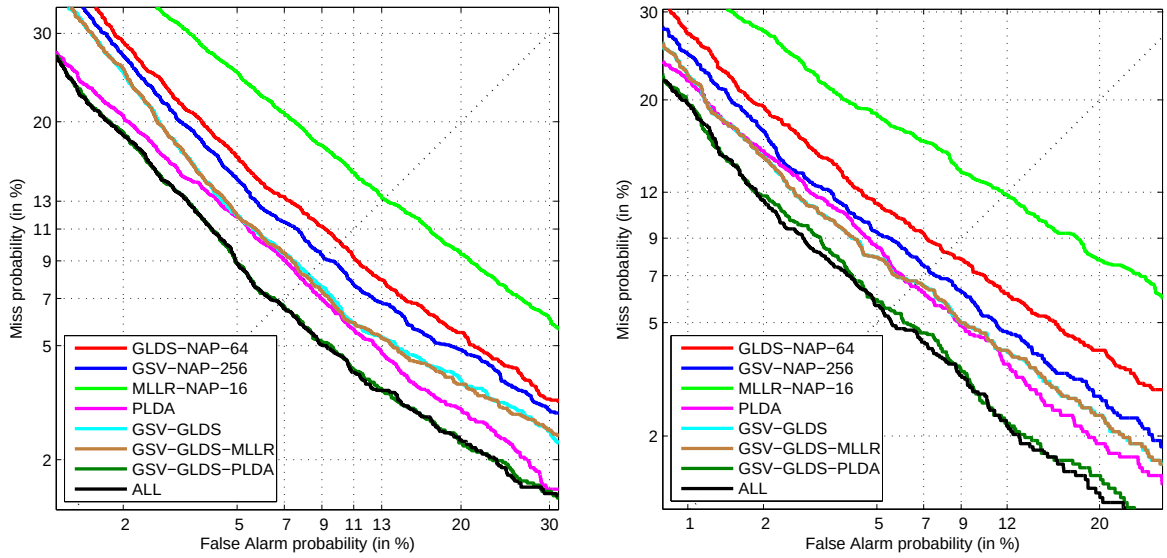

Figure 7.18: DET curves of individual and fused systems, results and fusion coefficients computed on NIST08. Left plot depicts the performance for females and the right plot for males.

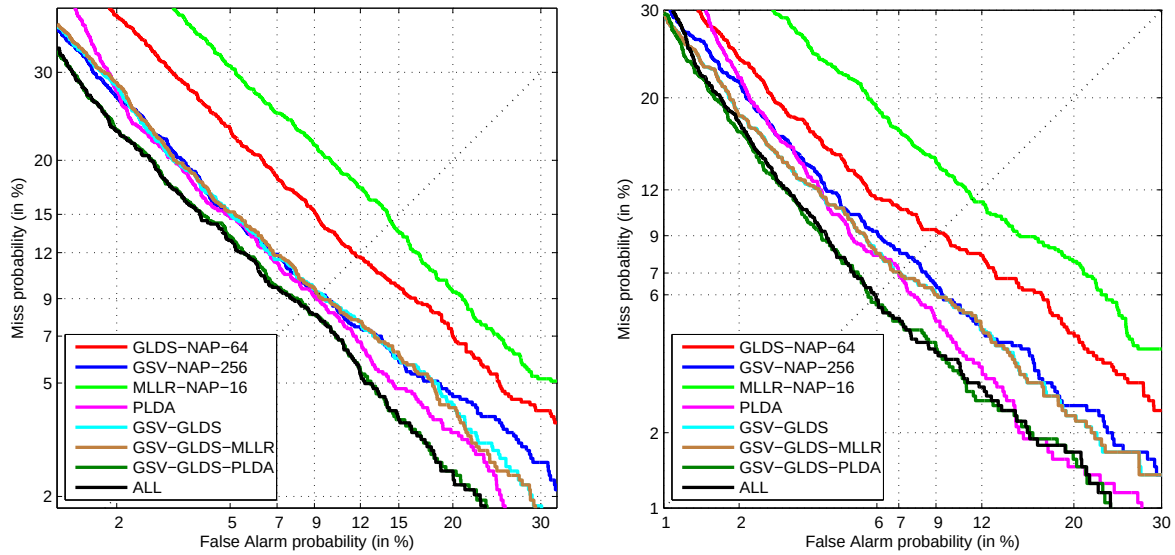

Figure 7.19: DET curves of individual and fused systems, fusion coefficients computed on NIST08, but results are given on NIST10. Left plot depicts the performance for females and the right plot for males.

Table 7.10: Error rates for different speaker recognition systems. Entries in the table represent EER[%]/minDCF obtained using each system independently on female and male speakers for NIST08 and NIST10.

	female		male	
	NIST08	**NIST10**	**NIST08**	**NIST10**
GSV-NAP-256	9.12/0.0465	9.31/0.0460	7.27/0.0343	7.68/0.0393
GLDS-NAP-64	10.12/0.0481	11.71/0.0569	8.21/0.0365	9.16/0.0430
MLLR-NAP-16	13.17/0.0552	14.38/0.0652	11.90/0.0442	11.37/0.0547
i-vec/PLDA	7.96/0.0377	9.12/0.0465	6.49/0.0314	7.05/0.0415
GSV-GLDS	8.23/0.0444	9.31/0.0469	6.65/0.0311	7.05/0.0377
GSV-GLDS-MLLR	8.14/0.0445	9.22/0.0471	6.70/0.0311	7.05/0.0377
GSV-GLDS-PLDA	6.78/0.0364	8.48/0.0423	5.51/0.0283	5.79/0.0358
ALL	6.78/0.0362	8.48/0.0423	5.30/0.0283	6.00/0.0364

7.9 Conclusion and Remarks

As proved the experiments none (variance) normalization of SVs in the SVM based system lead to the decrease of error rates. Conclusions were made that the covariance contained in the data does contain information and it should not be removed. Loosely speaking, the variance in distinct dimensions of SV can be "trusted". Such an observation is strongly supported by experiments on dimensionality reduction, which showed that directions with high variance (related to high values of eigenvalues) do contain much more information than those with small variance, and that a lot of redundant information is present in the SVM model (more precisely in the high dimensional parameter space of the SVM model). It would be of interest to try out also other types of normalization of feature vectors right after they were extracted to see if such conclusions hold.

Note that the SLK, MLLR and COV kernels in GSVs, MLLR- and GLDS-SVs are in fact simple linear kernel functions with pre-normalized SVs. The kernel function $K = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{x}$, where $\boldsymbol{x}$ is a type of SV (GSV/MLLR/GLDS) and $\boldsymbol{W}$ is a square (often diagonal) normalization matrix, can be replaced by $K = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}$, where $\boldsymbol{y} = \boldsymbol{W}^{1/2}\boldsymbol{x}$ are the pre-normalized SVs and $\boldsymbol{W}^{1/2}$ can be obtained using Cholesky decomposition, see (4.22).

Only a few techniques described in this thesis were also experimentally tested. One reason is the limited amount of time and computing resources, but in fact a lot of discussed methods was already examined in other published papers cited in this work. Methods analysed in this chapter are most commonly used in the field of speaker recognition and perform best on telephone speech (to some minor exceptions caused mainly by the lack of development data).

Experiments were performed on telephone speech conversations and on corpora containing thousands of speakers talking on different channels. Hence, methods used in the experiments along with the presented results relate to recognition of (channel distorted) telephone speech. Without doubts for another task with different environment conditions different methods may be more appropriate, e.g. on a "clean" data set a simple GMM based system may perform well.

A very common normalization technique is the T-norm or Z-norm (eventually their combination, or other similar alternatives [99]) presented in Section 5.6 used to normalize the verification score. There were attempts to apply such a score normalization, however the results did not get any better. Very good behaviour of a SVM based speaker recognition system without score normalization was observed also in [42, 100].

Chapter 8

Estimation of GMM Statistics on GPU

The Expectation-Maximization (EM) algorithm, in clustering often used also with Gaussian Mixture Models (GMMs), was in [101] identified as one of the top 10 data mining algorithms. GMMs trained via EM are widely used in many state-of-the art recognition and data mining systems. They are of most importance in the speaker recognition. They are utilized in the concept of supervectors (see Chapter 4) and Support Vector Machines (SVMs) and also in Factor Analysis (FA) based systems like JFA and i-Vectors (see Chapter 6). Another usage can be found in speech recognition systems based on Hidden Markov Models with output probabilities described by GMMs [102]. Nevertheless, GMMs are utilized also by biologists and immunologists for counting, sorting, and analyzing cells suspended in a fluid [103]. This chapter is based on the work published in [104].

All these techniques process huge amounts of data, thus demanding a superior computing power. Nowadays, parallel technologies like supercomputers, clusters, grids, and cloud infrastructures gain on importance [105]. A simpler option is to utilize the Graphics Processing Unit (GPU), which developed through time to a highly parallel and computationally powerful tool useful not only for graphics processing, but also for high performance computing [7]. The main advantage of GPUs over Central Processing Units (CPUs) is their price-performance ratio. Several manufacturers have put a lot of effort to improve their GPU's development environment in order to grant access to their GPU's computing power. This chapter focuses on NVIDIA's Compute Unified Device Architecture (CUDA). It should be stated, that implementations of the GPU algorithm may be easily included also into the above mentioned parallel technologies.

Focus is laid on GMMs described by diagonal covariance matrices. Only the estimation of GMM statistics is implemented on GPU, rather than the overall estimation of new GMM parameters. The statistics are more general and may be used also in other techniques, e.g. in the adaptation described in Section 3.2 or JFA discussed in Section 6.3. Once these statistics are available it is straightforward and fast to update the model parameters, see Section 8.2.

At first, basics of CUDA are described. Next, an efficient implementation of the estimation algorithm of GMM statistics on GPU is given. Also an augmented CPU version is proposed, it utilizes Streaming SIMD Extension (SSE) instructions, which make the estimation on CPU significantly faster. Note that the estimation process does not involve any approximations, GMM statistics obtained using any of the methods are equal (to some negligible rounding errors).

8.1 GMM Statistics

Assume a set of feature vectors $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, where $\dim(\boldsymbol{o}_t) = D$, and a GMM given by a set of parameters $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}_m\}_{m=1}^{M} = \{\omega_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}_{m=1}^{M}$ containing M Gaussians, their weights, mean

vectors and covariance matrices, respectively. Let us define a function

$$\mathcal{L}(\boldsymbol{o}_t, \{\boldsymbol{\lambda}_b, \ldots, \boldsymbol{\lambda}_e\}) = \log \sum_{m=b}^{e} \omega_m \mathcal{N}(\boldsymbol{o}_t | \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \tag{8.1}$$

where $b \geq 1, e \leq M$. Only diagonal covariance GMMs will be assumed, where $\boldsymbol{\sigma}_m^2 = \mathrm{diag}(\boldsymbol{\Sigma}_m)$, hence

$$\mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda}_m) = \log(\omega_m) + \sum_{i=1}^{D} \log \mathcal{N}(o_{t,i} | \mu_{m,i}, \sigma_{m,i}^2), \tag{8.2}$$

where $\mathcal{N}(o|\mu, \sigma^2)$ denotes the Gaussian kernel function with mean μ and variance σ^2. Recall the statistics from Section 3.1:

$$\gamma_m(\boldsymbol{o}_t) = \exp\left(\mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda}_m) - \mathcal{L}\left(\boldsymbol{o}_t, \{\boldsymbol{\lambda}_k\}_{k=1}^M\right)\right) \tag{8.3}$$

$$c_m = \sum_{t=1}^{T} \gamma_m(\boldsymbol{o}_t), \tag{8.4}$$

$$\boldsymbol{\varepsilon}_m = \textstyle\sum_{t=1}^{T} \gamma_m(\boldsymbol{o}_t)\boldsymbol{o}_t, \quad \boldsymbol{\varepsilon}_m^2 = \sum_{t=1}^{T} \gamma_m(\boldsymbol{o}_t)\boldsymbol{o}_t\boldsymbol{o}_t^{\mathrm{T}} \tag{8.5}$$

are the m^{th} Gaussian's posterior probability given a feature vector $\boldsymbol{o}_t$, the m^{th} Gaussian's soft count (the zero moment) and the (*unnormalized*) first and second moment of feature vectors aligned to Gaussian m, respectively. Note that $\mathcal{L}(\boldsymbol{o}_t, \{\boldsymbol{\lambda}_m\}_{m=1}^M) = \mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda})$ represents the log-likelihood of $\boldsymbol{o}_t$ given the model $\boldsymbol{\lambda}$.

8.2 EM and Adaptation

Expectation Maximization (EM) algorithm is an iterative procedure, where initial parameters $\boldsymbol{\lambda}^0$ have to be given. For each iteration the increase in the log-likelihood of the feature vectors given the model parameters is guaranteed. Thus, $\sum_t \mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda}^k) \geq \sum_t \mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda}^{k-1})$, where the upper index k denotes the iteration number. In each iteration statistics (8.3) – (8.5) are computed with old values of model parameters, and then they are used to update the model according to:

$$\bar{\omega}_m = \tfrac{c_m}{T}, \quad \bar{\boldsymbol{\mu}}_m = \tfrac{1}{c_m}\boldsymbol{\varepsilon}_m, \quad \bar{\boldsymbol{\Sigma}}_m = \tfrac{1}{c_m}\boldsymbol{\varepsilon}_m^2 - \bar{\boldsymbol{\mu}}_m\bar{\boldsymbol{\mu}}_m^{\mathrm{T}}, \tag{8.6}$$

where $\bar{\boldsymbol{\lambda}}$ are the new GMM parameters. Moreover, note that the same statistics are used also in all the adaptation techniques described in Chapter 3, in the extraction process of GMM based supervectors (4.5), (4.7), and also when extracting the JFA, i-vector related supervectors (6.80) and (6.106), respectively.

8.3 Estimation Utilizing CUDA

GPU's CUDA may be seen as a fully parallel system operating with hundreds of threads at once. According to the GPU architecture threads are organized into *thread blocks*. Thread blocks are independent of each other (algorithm executed in each of the blocks does not depend on what is going on in other blocks), while threads in each block are allowed to cooperate. All thread blocks execute the same algorithm called a *kernel*. Note that not only threads in a block, but also several thread blocks may be executed at once. Hence, CUDA parallelism is provided at 2 levels - threads and block of threads. All thread blocks are ordered in an one- or two-dimensional *grid* (a 2 dimensional grid

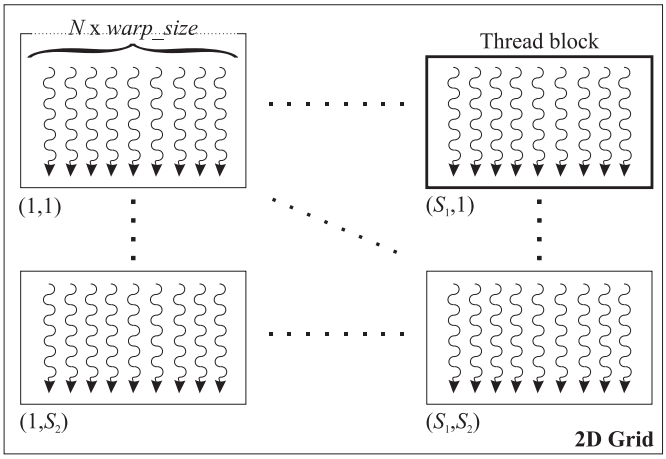

Figure 8.1: Two dimensional $S_1 \times S_2$ grid with thread blocks. Each thread block contains several threads, where the optimal number of threads is a multiple of the warp size.

is depicted in Fig. 8.1). Each thread block carries specific information about its position within the grid (row and column position of the block in the grid).

A high GPU computing performance can be fully utilized only with proper memory management. Several memory types exist, which significantly differ in their size, access speed and access permission. *Global memory* (GM) has read/write access, has hundreds of mega bytes available and can be accessed from every block and every thread, but the access latency is relatively high. The best performance of GM can be achieved using the *Texture Memory* (TM). TM can be seen as a part of GM, but it is read only and cached, thus the access speed may be significantly faster than in the case of GM. Another type of memory is the *Shared Memory* (SM), which storage size is around kilo bytes, but the access speed is very high (very low latency). SM is visible only for threads in a thread block. In summary, one has to carefully choose the memory management according to a given task.

For further details and deeper understanding of the problem the reader is referred to [106].

8.3.1 Preparing the Data

In order to make the best of the GPU computing power one has to align the data into Memory-Aligned-Blocks (MABs). The optimal size of a MAB is closely related to the number of threads in a thread block. Number of threads in a block is user dependent, but optimally has to be a multiple of the *warp size*. Warp size is hardware dependent and represents the minimum number of threads in a thread block that run at once (mostly a multiple of 32) – *run in a warp*. For the best performance threads in a warp have to access data in the memory sequentially therefore data in MABs have to be properly organized.

We have input data (a set of feature vectors $\boldsymbol{O}$ and a set of GMM parameters $\boldsymbol{\lambda}$), temporary data (Gaussian posteriors (8.3) together with log-likelihoods of feature vectors given $\boldsymbol{\lambda}$), and output data (first and second moments (8.5) and soft counts (8.4)) that have to be properly organized in the GPU memory. The memory storage of feature vectors, model means and Gaussian posteriors are depicted in Fig. 8.2, Fig. 8.3 and Fig. 8.4, respectively. Storage of GMM diagonal variances is the same as the storage of model means depicted in Fig. 8.3. The reason why model parameters and Gaussian posteriors are stored in group of 4 is that CUDA supports *X4* data types (e.g. *short4, int4, float4*, etc.) – one can read data from the memory in quaternions.

Rather than to store only weights of Gaussians we store the precomputed normalization coefficient

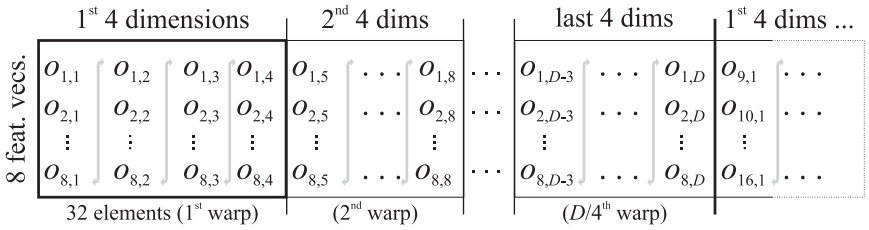

Figure 8.2: Organization of feature vectors $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$ in the GPU's global memory, where $\dim(\boldsymbol{o}_t) = D$. Data are stored column-wise – 1st dimension of first 8 feature vectors then 2nd dimension of first 8 samples, etc. In each warp a block of memory is read sequentially enabling optimal speed performance.

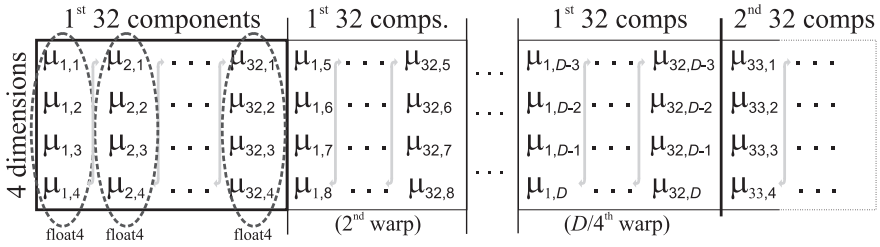

Figure 8.3: Organization of model means $\boldsymbol{\mu}_i$ in the GPU's global memory. Storage of GMM diagonal variances $\boldsymbol{\sigma}_i^2$ is the same. Data are stored column-wise – 1st four dimensions of $\boldsymbol{\mu}_1$ then 1st four dimensions of $\boldsymbol{\mu}_2$, etc.

of each Gaussian, its logarithm is given as

$$g_m = \log(\omega_m) - 0.5D\log(2\pi) - 0.5\log|\boldsymbol{\Sigma}_m|\,. \tag{8.7}$$

Memory management of g_m, soft counts (8.4) and first and second moments (8.5) is trivial (recall that only the diagonal of second moments is stored), they are all stored sequentially in ascending order according to the number of Gaussian they belong to (e.g. a vector of first moments $[\boldsymbol{\varepsilon}_1^{\mathrm{T}}, \boldsymbol{\varepsilon}_2^{\mathrm{T}}, \ldots, \boldsymbol{\varepsilon}_M^{\mathrm{T}}]$ represents one memory block). Also data log-likelihoods $\mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda})$ are stored sequentially in ascending order according to the position of a vector $\boldsymbol{o}_t$ in the set $\boldsymbol{O}$, thus forming a vector $[\mathcal{L}(\boldsymbol{o}_1, \boldsymbol{\lambda}), \ldots, \mathcal{L}(\boldsymbol{o}_T, \boldsymbol{\lambda})]$.

It should be stated that all the GPU's memory management of feature vectors, model parameters, temporary data (once computed) is assigned to the cached TM (all data are visible to all thread blocks and their threads). However, feature vectors are copied to the faster SM in some kernels, see next section.

8.3.2 CUDA Kernels

Kernels specify what should threads in a thread block do (number of threads is specified by the user) assuming additional information about the position of a thread block in a grid, grid dimension and given input data. The position information along with the grid dimension is utilized to properly divide input data into smaller independent portions. Each of the data portions is then handled by a separate thread block according to the specified kernel function.

Not all the tasks can be parallelized using only one kernel function since a problem can not always be divided into several fully independent parallel subtasks. More often a result of one subtask depends on a result of a different subtask. However, such tasks may have only a few points where they need to exchange their outcomes. Thus, to parallelize the task one has to employ more kernels. We have proposed 4 kernels

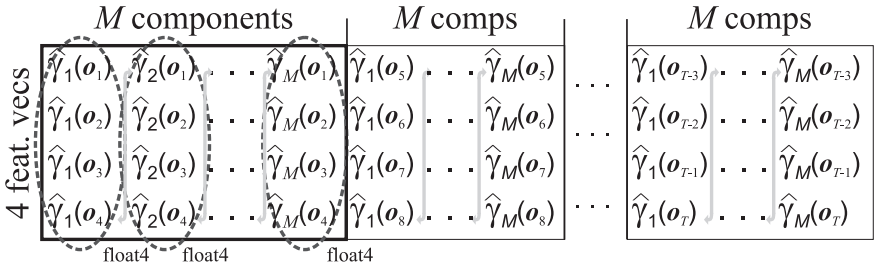

Figure 8.4: Organization of unnormalized posteriors $\hat{\gamma}_m(\boldsymbol{o}_t) = \mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda}_m)$ given in (8.2) of a Gaussian in the GPU's global memory. Data are stored column-wise – 1st four posteriors of 1st Gaussian given first four feature vectors $\{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_4\}$, 1st four posteriors of 2nd Gaussian, etc.

- $\hat{\gamma}$*-kernel* – computes $\hat{\gamma}_{m,t} = \mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda}_m)$ for each t, m,
- $\mathcal{L}$*-kernel* – computes overall log-likelihood $\mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda})$ for each t,
- γ*-kernel* – normalizes each $\hat{\gamma}_{m,t}$ by $\mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda})$ to get a proper Gaussian posterior (8.3),
- ε*-kernel* – estimates first and second moments $\boldsymbol{\varepsilon}_m, \boldsymbol{\varepsilon}_m^2$ for each m.

In order to describe the data portions handled by distinct kernels described in the next section assume a set $\boldsymbol{\Gamma} = \{\{1, \ldots, Q_T\}, \ldots, \{T - Q_T, \ldots, T\}\} = \{\boldsymbol{\Gamma}_i\}_{i=1}^{S_1}$ containing equally large disjoint subsets of feature vector indexes, a set $\boldsymbol{\Omega} = \{\{1, \ldots, Q_M\}, \ldots, \{M - Q_M, \ldots, M\}\} = \{\boldsymbol{\Omega}_j\}_{j=1}^{S_2}$ containing equally large disjoint subsets of indexes of GMM Gaussians, and a set $\boldsymbol{\Delta} = \{\{1, \ldots, Q_D\}, \ldots, \{D - Q_D, \ldots, D\}\} = \{\boldsymbol{\Delta}_d\}_{d=1}^{S_3}$ formed by equally large disjoint subsets of dimension indexes of feature vectors. Q_T, Q_M and Q_D are user defined scalars, where $Q_T \leq T$, $Q_M \leq M$ and $Q_D \leq D$. Loosely speaking, $\Gamma_{i,j} = (i-1) \cdot Q_T + j$, $\Omega_{i,j} = (i-1) \cdot Q_M + j$, $\Delta_{i,j} = (i-1) \cdot Q_D + j$. Memory management depicted in Figs. 8.2 and 8.3 is well suited for $Q_T = 8$, $Q_M = 32$, $Q_D = 4$. In order to preserve the robustness of calculations all intermediate results are kept in logarithms (as long as possible).

$\hat{\gamma}$-kernel operates on a two-dimensional $S_1 \times S_2$ grid, which rows indicate the portion of feature vectors and the columns of the grid indicate the portion of Gaussians to be processed. Hence, the $(i,j)^{th}$ thread block operates with sets $\boldsymbol{\Gamma}_i$ and $\boldsymbol{\Omega}_j$, and the output of the thread block are the corresponding weighted log-likelihoods $\hat{\gamma}_{m,t} = \mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda}_m)$ of a Gaussian written to given positions in GM as illustrated in Fig. 8.4. At the very beginning of the kernel execution, the complete input portion of feature vectors $\{\boldsymbol{o}_t\}_{t \in \boldsymbol{\Gamma}_i}$ (in Fig. 8.2 are these all the dimensions of 8 feature vectors) handled by one thread block is read sequentially from GM and written to SM. Each thread estimates one Gaussian's weighted log-likelihood of Q_T different feature vectors, thus a set $\{\hat{\gamma}_{m,t}\}_{t \in \boldsymbol{\Gamma}_i}$ for one specific m. Particular steps of the kernel algorithm are described in Alg. 5. The sum $\sum_{x \in \boldsymbol{\Delta}_d}(\ldots)$ across a subset of dimensions of a feature vector and across a subset of dimensions of GMM parameters in the for-loop is caused by the fact that the model parameters are read from TM as *float4*, thus $\boldsymbol{\Delta}_d = \{(d-1) \cdot 4 + j\}_{j=1}^{4}$ consists of 4 indexes of 4 dimensions (see Fig. 8.3). The relationship between the storage of model parameters and feature vectors should be now clearer – mainly the reason why the vectors are divided to dimension blocks of 4. Also note that rather than using the for-loop through indexes in $\boldsymbol{\Gamma}_i = \{\Gamma_{i,1}, \ldots, \Gamma_{i,Q_T}\}$ we unroll the loop in order to boost the performance.

$\mathcal{L}$-kernel is a sum kernel, it computes the overall log-likelihood of each feature vector given a GMM. Hence, the input to the kernel is the output of the $\hat{\gamma}$-kernel. The output of $\mathcal{L}$-kernel is a

Algorithm 5 $\hat{\gamma}$-kernel function $\rightarrow$ blocks $\mathbf{\Gamma}_i, \mathbf{\Omega}_j$

Require: Thread block position (i, j) in the grid

SM $\overset{\{\boldsymbol{o}_t\}_{t\in\mathbf{\Gamma}_i}}{\longleftarrow}$ GM
$m := \Omega_{j,thread\_index}$
$\hat{\gamma}_{m,\Gamma_{i,1}} := g_m; \ldots; \hat{\gamma}_{m,\Gamma_{i,Q_T}} := g_m$
for $d = 1$ to S_3 **do**
  for $t \in \mathbf{\Gamma}_i$ **do**
    $\hat{\gamma}_{m,t} := \hat{\gamma}_{m,t} + \sum_{x\in\mathbf{\Delta}_d}(o_{t,x} - \mu_{m,x})^2/\sigma^2_{m,x}$
  end for
end for
GM $\longleftarrow \{\hat{\gamma}_{m,\Gamma_{i,k}}\}_{k=1}^{Q_T}$

Algorithm 6 ε-kernel function $\rightarrow$ blocks $\mathbf{\Omega}_i, \mathbf{\Delta}_j$

Require: Thread block position (i, j) in the grid

$m := \Omega_{i,thread\_index}$
$c_m := 0$
$\varepsilon_{m,\Delta_{j,1}} := 0; \ldots; \varepsilon_{m,\Delta_{j,Q_D}} := 0$
$\varepsilon^2_{m,\Delta_{j,1}} := 0; \ldots; \varepsilon^2_{m,\Delta_{j,Q_D}} := 0$
for $q = 1$ to S_1 **do**
  SM $\overset{\{o_{t,d}\}_{t\in\mathbf{\Gamma}_q, d\in\mathbf{\Delta}_j}}{\longleftarrow}$ GM
  for all $t \in \mathbf{\Gamma}_q$ **do**
    $c_m := c_m + \gamma_m(\boldsymbol{o}_t)$
    for all $d \in \mathbf{\Delta}_j$ **do**
      $\varepsilon_{m,d} := \varepsilon_{m,d} + \gamma_m(\boldsymbol{o}_t) \cdot o_{t,d}$
      $\varepsilon^2_{m,d} := \varepsilon^2_{m,d} + \gamma_m(\boldsymbol{o}_t) \cdot o^2_{t,d}$
    end for
  end for
end for
GM $\longleftarrow \{\varepsilon_{m,\Delta_{j,k}}, \varepsilon^2_{m,\Delta_{j,k}}\}_{k=1}^{Q_D}$
GM $\longleftarrow c_m$

set $\{\mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda})\}_{t=1}^{T}$ written to the GM as described in Section 8.3.1. Several efficient algorithms for a parallel sum have already been proposed, we use the implementation described in [107].

γ-kernel performs the normalization of each $\hat{\gamma}_{m,t}$ with $\mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda})$ and produces true posteriors of GMM Gaussians, thus $\gamma_m(\boldsymbol{o}_t) = \exp\left(\hat{\gamma}_{m,t} - \mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda})\right)$. These are written to the same positions as their unnormalized counterparts. One thread block processes Q_T feature vectors from a set $\mathbf{\Gamma}_i$ and all the Gaussians, and outputs $\{\gamma_m(\boldsymbol{o}_t)\}_{t\in\mathbf{\Gamma}_i, m\in\mathbf{\Omega}}$.

ε-kernel operates on a two-dimensional $S_2 \times S_3$ grid, which rows indicate the portion of Gaussians and the columns of the grid indicate the portion of feature dimensions to be processed. Hence, the $(i, j)^{th}$ thread block operates with sets $\mathbf{\Omega}_i$ and $\mathbf{\Delta}_j$, and the output of a thread block are the dimension blocks of first and (diagonal) second moments (8.5) of features aligned to a given Gaussian along with the soft counts (8.4). The output is written to GM on positions described in Section 8.3.1. Thus, each thread block processes the whole set of feature vectors, however only values for a specific subset of dimensions of first and second moments are estimated. The ε-kernel operates with all the data –

feature vectors, model parameters and temporary data obtained as the output of the γ-kernel. The kernel algorithm is described in Alg. 6. Note that $\{o_{t,d}\}_{t\in\mathbf{\Gamma}_q, d\in\mathbf{\Delta}_j}$ is one block depicted in Fig. 8.2 containing 8 feature vectors ($Q_T = 8$) and their 4 dimensions ($Q_D = 4$), thus 32 elements in common. Again, to boost the performance instead of using the most inner for-loops through indexes in $\mathbf{\Gamma}_q, \mathbf{\Delta}_j$ we unroll the loops. Also note that posteriors $\boldsymbol{\gamma}_m(\boldsymbol{o}_t)$ are read from TM as *float4* data types. Such set up is efficient mainly in cases when the number of Gaussians in the GMM or dimension of feature vectors are high, otherwise only a few thread blocks have to be executed what decreases the speed performance. In these cases the input portion of feature vectors is divided into Q_N blocks and each thread block accumulates statistics for one of these blocks. Hence, now the ε-kernel processes T/Q_N feature vectors, Q_M Gaussians and Q_D dimensions. After all the statistics for disjoint feature sets have been accumulated an additional kernel is utilized in order to sum up the resulting Q_N distinct statistics.

In summary, one has to prepare the data on CPU and upload them to GPU's GM, where further management is needed in order to represent the data in TM. Next, $\hat{\gamma}$-kernel is launched, which computes the unnormalized Gaussian posteriors. These serve as input into the likelihood $\mathcal{L}$-kernel. Subsequently, Gaussian posteriors are normalized running the γ-kernel. At last, all the data are utilized in the ε-kernel, which estimates first and second moments and soft counts of each Gaussian. Results are written in the GM, thus they have to be copied back to CPU's memory and reorganized according to the user's needs.

8.4 Estimation Utilizing SSE

We have tried to speed-up also the estimation on CPU utilizing Streaming SIMD Extensions (SSE), where SIMD stands for Single Instruction, Multiple Data. The power of SSE is that it can perform several instructions (addition, subtraction, multiplication, etc.) at once using 128-bit registers. Thus, assuming 32-bit single-precision floating point (SPFP) numbers one can perform 4 operations at a time.

We have incorporated the SSE instructions into the estimation of $\mathcal{L}(\boldsymbol{o}_t, \boldsymbol{\lambda}_m)$ given in (8.2), which is the most frequent, thus most time consuming operation. More precisely, SSE is used when computing the exponential part of the normal distribution $\sum_{d=1}^{D}(o_d - \mu_d)^2/\sigma_d^2$. Using SSE and SPFP such sum can be added up in $D/4$ steps. In situations where D is not a multiple of 4 one has to correctly align the memory (pad ends with zeros) where GMM parameters (means and variances) and feature vectors are stored. Additional less significant speed bursts may be acquired extending the SSE instructions into the accumulation process of moments given in (8.5).

8.5 Experiments

Experiments were performed on a single EM iteration. Data were taken from NIST SRE 2008, only training data were used for adaptation. More precisely, it was the short2 training condition and only male telephone speech of approximately five minutes total duration was used (non-speech events were discarded during feature extraction). In common 648 speakers were involved, approximately 54 hours of speech were used. In summary, the training data consisted of 3,125,506 (3125.6k) feature vectors of dimension 40.

In our implementation we used only floating point arithmetic. The user defined constants used in Section 8.3.2 were set to $Q_T = 8$, $Q_M = 32$, $Q_D = 4$ (such settings correspond with Figs 8.2-8.4), and $Q_N = 8$. The number of threads in each thread block was set to 32.

CPU and SSE implementations were tested on 2.39 GHz Intel 4 GB RAM PC, the GPU imple-

mentation was tested on low-end NVIDIA GeForce GTX 280 video card and the algorithms were developed in CUDA toolkit 3.1. All the GPU time consumptions were computed as the sum of times of all the executed kernels.

8.5.1 Analysis of the Implementation Performance

Comparison of time consumptions of the proposed implementation can be found in Tab. 8.1. Only the time needed to accumulate statistics was measured, and just one thread was used in all CPU implementations. Results are given in seconds, the data set consists of 3125.6k feature vectors of dimension 40. GPU is approximately 150 times faster than the CPU-SSE implementation and more than 400 times faster than the naive CPU implementation. The relative GPU execution times of particular kernels can be found in Fig. 8.5.

The comparison of GPU and CPU version strongly depends on the implementation of the CPU version. In order to express only the performance of the GPU implementation one can evaluate the number of floating point operations per second (FLOPS). Counts of operations executed in each kernel are

- $\hat{\gamma}$-*kernel* – $4 \times D \times T \times M$ operations,
- $\mathcal{L}$-*kernel* – $T \times M$ of logarithmic addition functions (which we rated as 13 operations), thus $T \times M \times 13$ operations,
- γ-*kernel* – $T \times M$ of 5 simple operations + one exponential operation (equal to 4 simple operations), thus $T \times M \times 9$
- ε-*kernel* – $4 \times D \times T \times M + T \times M$ operations.

We distinguish simple operations as addition, subtraction, and multiplication from operations as logarithms and exponentials, which are on GPUs calculated using special function unit that have four times lower throughput. Therefore we rate logarithms and exponentials as 4 simple operations. Hence, when number of Gaussians M, number of feature vectors T, and dimension of feature vectors D are known, the overall number of operations of the estimation can be computed. In our case this is the sum of operations of all 4 kernels. Tab. 8.2 contains the number of operations per second (Giga FLOPS = GFLOPS), which are computed as the number of operations needed to estimate GMM statistics for various number of Gaussians divided by the estimation times from Tab. 8.1. GFLOPS range from 163.4 to 242.1 because larger models utilize GPU cores better, and the overhead of kernel executions is relatively lower in cases of larger models. The theoretical peak of the GTX 280 GPU is 933 GFLOPS (according to specifications of the manufacturer), which is in comparison to the performance on real tasks significantly overstated. In a benchmark task performed in [108], where a well optimized task of multiplication of two large matrices on GTX 280 GPU is carried out, the achieved performance varies between 190 – 375 GFLOPS in dependence on matrix sizes. Hence, GFLOPS of our implementation are in the range of such a well optimized task. Nevertheless, matrix-matrix multiplications consist only of fused multiplication/addition instructions, which are evaluated in a single GPU clock (doubles the GFLOPS performance). However, our task consists also from other instructions, which are not as efficient.

8.5.2 Comparison with Previous Works

We have tried to compare the time consumptions also with other implementations. We have tested several freely available implementations, but all of them failed (lack of numerical stability) on

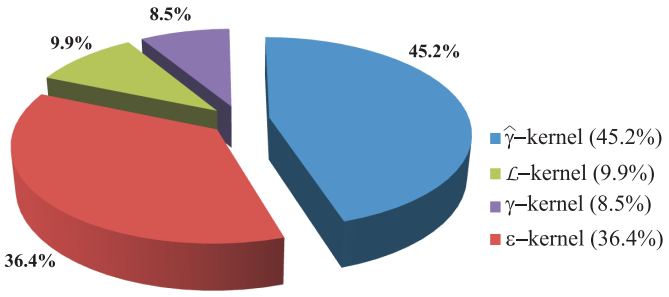

Figure 8.5: Relative GPU execution times for all of the kernels described in Section 8.3.2.

Table 8.1: The amount of time in seconds needed to estimate statistics of 3125.6k feature vectors of dimension 40 for various number of GMM Gaussians.

#Gauss.	32	64	128	256	512	1024	2048
CPU	78	150	287	559	1094	2174	4289
SSE	32	54	95	171	325	619	1199
GPU	0.21	0.32	0.63	1.18	2.34	4.57	9.07

Table 8.2: Performance of the algorithm running on NVIDIA GeForce GTX 280 in GFLOPS when processing 3125.6k feature vectors of dimension 40 for various number of GMM Gaussians.

#Gauss.	32	64	128	256	512	1024	2048
GFLOPS	163.4	214.4	217.8	232.6	234.6	240.2	242.1

Table 8.3: Comparison given in milliseconds of different implementations for different amounts of training data assuming feature vectors of dimension 32 and GMM with 32 Gaussians.

#samples	Kumar et al.	A.Pangborn	UWB	MATLAB®
153.6k	215.0	51.1	9.25	10936.0
230.4k	264.9	71.1	13.99	16461.0

our large dataset of high dimensional real data. We have found two recent publications (worth to be mentioned) interested in the GPU implementation of the EM algorithm focusing on GMMs with diagonal covariances, namely a publication by Kumar et al. [109] and a master thesis from Andrew Pangborn [103].

Experiments performed by Kumar et al. used NVIDIA Quadro FX 5800, which is almost identical to the NVIDIA TESLA C1060 on which the experiments of Andrew Pangborn were performed, and to NVIDIA GeForce GTX 280 on which our experiments were performed. Time consumptions of both implementations were taken from Tab. 5.8 from [103]. In order to compare the implementations to ours we set up same conditions as in [109] and [103]. Hence, we reduced the dimension of our data to 32 and took only 153.6k and 230.4k feature vectors. Tab. 8.3 is the extended table containing also our results denoted as UWB and the CPU reference computed in MATLAB® 7.5.0.342 (R2007b) utilizing the Statistics Toolbox function `gmdistribution.fit()` (only the time spent on estimation of statistics was measured).

As can be seen from Tab. 8.3, the implementation proposed in this chapter outperformed the others. It is more than 5 times faster than A. Pangborn's implementation and more than 20 times faster than Kumar's implementation.

The key part of the speed up is the proper memory management of the data adhering to the rules of coalesced access [106]. In addition, data loaded to the kernels are reused as much as possible (higher degree of parallelization), e.g. the log-likelihoods are estimated for several feature vectors and several Gaussians at once in each kernel, the same principle holds for the accumulation kernel (see

descriptions of $\hat{\gamma}$- and ε-kernel in Section 8.3.2). Another important performance related technique lays in the use of the Texture Memory (TM) with *float4* data types for read-only data. Data shared across a thread block or data that are accessed repeatedly should be copied into the Shared Memory (SM) in advance. The mentioned advices are of course well known, but it is quite difficult to integrate them to a specific task.

Another drawback of Kumar et al. implementation is that the GPU memory requirements are very high, this holds particularly also for A. Pangborn. The most of the memory is occupied by the intermediate results. Since the statistics are additive, the computation can be divided into smaller parts that require significantly lower amount of memory. In our case the most of the memory is occupied only by the input data, thus we are able to fit up to 6 millions of 40 dimensional feature vectors into the GPU memory of size 1 GB. However, even in cases where a huge data set containing several hundreds of millions of feature vectors needs to be processed, the memory problem can be solved efficiently. Most of GPUs dispose of concurrent copy and execution feature, thus additional data can be uploaded to the GPU memory while already uploaded data are processed.

8.6 Conclusion and Remarks

Since the EM algorithm does converge only locally it is often convenient to run EM several times with different initializations. Hence, in order to train a reliable GMM via EM one has to perform a lot of reestimations. With increasing amount of training data and increasing complexity of models, the training of GMMs becomes very time consuming. As has been shown (see Tab. 8.1), the GPU implementation offers a huge increase in the speed of GMM training. However, the final speed up strongly depends not only on the GPU hardware, but also on a proper implementation itself (see Tab. 8.3). The estimation process can be easily parallelized also on the CPU (e.g. dividing feature vectors to smaller disjoint sets, estimating statistics for each set, and at the end adding the statistics up), but the resources spent on the hardware are much higher than in the case of GPU, which is parallel inherently.

We have focused on the estimation of GMM statistics with diagonal covariances since often full covariance GMMs can be accurately replaced by diagonal covariance GMMs with higher number of Gaussians. The estimation of diagonal covariances is more robust mainly with increasing dimension of feature vectors. This is often the case in tasks of speech and speaker recognition, where frequently only the diagonal covariances are used. Finally, note that one of the outputs of the algorithm produced by the $\mathcal{L}$-kernel is also the log-likelihood of feature vectors given a GMM.

Chapter 9

Conclusion and Future Work

Techniques of automatic speaker recognition developed in the last decade were presented. The main emphasize was laid on the modelling of feature vectors once extracted. Feature vectors are of relatively small dimension (in our case the dimension was 40) and are based on the power spectrum of the speech signal. Feature vectors are mapped to a high dimensional space with the use of a generative model. The task of the generative model is to segment the feature space according to the concentration of feature vectors from the development set localized in specific areas of the feature space. The high dimensional vectors denoted Supervectors (SVs) are composed of various statistics related to feature sets of a speaker, and they can be thought of as a higher level features. For each recording of a speaker only one SV is extracted. Since nowadays large corpora containing hundreds of speakers recorded on several channels (we say that several sessions of a speaker are at hand) are available, information on channel distortions is utilized in order to identify directions in the SV space most responsible for channel and speaker changes. Moreover, since SVs are of substantially high dimension these techniques incorporate also the dimensionality reduction.

In order to give a general overview of the topics investigated in the last decade also less efficient techniques were presented and analysed. Even if these techniques do not contribute to the performance of a speaker recognition system in specific environment conditions (telephone speech), what can be caused e.g. by the use of some normalization techniques in the extraction process as discussed in Section 7.6.1, they can be applied with a success to another classification problem.

The thesis was devoted to a thorough description of methods used in the experiments in a logical sequence. Following problems were solved:

1. *An efficient implementation of the EM estimation algorithm on a Graphics Processing Unit (GPU).* More precisely, the evaluation of EM related statistics was transfered to GPU. Since several thousand hours of speech had to be processed yielding over 10^4 millions of 40 dimensional feature vectors, and since these statistics are required not only when estimating the Universal Background Model (UBM), but are needed also when extracting SVs used with Support Vector Machines (SVMs) and in the i-vector extraction, the speed up was of great importance. Moreover, the EM algorithm and GMM estimation are frequently used also in other fields than the speaker recognition [103, 101].

2. *The influence of normalizations of SVs on the performance of the SVM system.* Surprisingly, the lowest error rates were acquired without any normalizations of SVs. The result was attributed to the pre-normalization of low dimensional feature vectors using the Feature Warping (FW) technique.

3. *An efficient implementation of the PLDA estimation algorithm.* In order to investigate the

impact of development data sets on the PLDA modelling analysed in Section 7.7 and performance of the speaker verification system more than 4000 PLDA models were trained for 800 dimensional i-vectors (each training consisted from 50 iterations).

4. *Relation of Nuisance Attribute Projection (NAP) to Factor Analysis (FA) based channel compensation.* Since NAP used with SVM does a channel compensation and so does the FA model used in JFA/PLDA, the similarities and working principles of both approaches were examined. Both problems were converted to the formulation of Least Squares (LS) and reviewed in the new light of LS yielding interesting conclusions on handling the noise. In simple terms, both do the eigenvector decomposition of a within-speaker covariance matrix, but FA based approach does in addition scale the directions according to the estimated noise level.

5. *The influence of the size of development sets and fusion of PLDA decompositions of several total variability spaces (composed from i-vectors from individual development corpora) on the performance of the speaker verification.* It was shown that if enough data to train a reliable PLDA model are available then it is more convenient to train one PLDA model for each development corpus, and let a fusion algorithm assign a weight to each verification score related to such a model. If variations in one corpus would be much higher than in other corpora, and in addition data from such a corpus would be inappropriate for given recognition conditions (e.g. telephone speech), the PLDA model trained from pooled corpora could notably spoil the recognition.

6. *Information redundancy in SVM models.* In addition to the dimensionality reduction of SVs via i-vector extraction, also the dimensionality reduction of SVM model was investigated. Obviously, SVM model does contain a lot of redundancies yielding a SVM decision hyperplane of much lower dimension. Such an observation may be of help when proposing a kernel function, which could instead of mapping to higher dimensions utilize some (e.g. non-linear) mapping to a lower dimensional space.

7. *Complementarity of discussed methods.* It was shown that methods used in the experiments do possess a complementary information since the fused system outperformed all systems based on particular methods.

9.1 Future Work

Experiments in [6] have shown the contribution of full covariance matrices used in GMMs in comparison to diagonal ones when dealing with i-vectors. Computation costs associated with full covariances are huge, therefore the GPU implementation should be extended to handle also full covariance GMMs.

As can be noticed, by now linear methods dominate the task of speaker recognition. However, recently non-linear methods based on Gaussian Processes (GPs) [111] such as Gaussian Process Latent Variable Model (GPLVM) [112], or Neural Networks for speech recognition [113, 114] are developed and widely expanded. Hence, it would be of interest to investigate the possibilities of these methods also in the task of speaker recognition. E.g. the PLDA model in the i-vector space could be replaced by a non-linear model: the total variability space could be chosen of higher dimension and the final reduction could be obtained utilizing a non-linear method. Also a non-linear SVM kernel could be used, but instead of transforming to a higher dimension, it would transform the SVs to much lower dimension utilizing non-linearities – idea follows experiments given in Section 7.6 on dimensionality reduction of SVM models. Even the UBM step could be replaced by a well chosen non-linear transformation of feature vectors to some other space yielding again one vector of higher fixed dimension for

each recording. A lot of possibilities are available and have to be examined concerning the non-linear mappings.

Finally, lot of development sets of spoken speech are available, but there is none method, which would be able to pick suitable data for a specific task. The question whether more development data irrelevant on their source are always better is still unanswered. Therefore, more experiments aimed to reveal the influence of distinct corpora in different acoustic environments on the task of speaker recognition would be of significance. Also an analysis of the influence of development data on the performance of individual methods, in order to guarantee robustness of these methods under varying conditions, would be of interest.

Appendix A

First Appendix

We are going to prove that for two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ both of size $D \times D_p$ and both of full column rank, which columns span the same subspace, the projection matrices

$$\begin{aligned} \boldsymbol{P}_A &= \boldsymbol{A}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}, \\ \boldsymbol{P}_B &= \boldsymbol{B}(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B})^{-1}\boldsymbol{B}^{\mathrm{T}} \end{aligned} \tag{A.1}$$

equal. Denote $\boldsymbol{Q}$ a matrix of size $D \times D_p$ spanning the same subspace as $\boldsymbol{A}$ and $\boldsymbol{B}$, but in addition $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \boldsymbol{I}$ (columns of $\boldsymbol{Q}$ are orthonormal). Now, we can express both matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ in terms of $\boldsymbol{Q}$ as

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{G}_A, \;\; \boldsymbol{B} = \boldsymbol{Q}\boldsymbol{G}_B, \tag{A.2}$$

where $\boldsymbol{G}_A, \boldsymbol{G}_B$ are of size $D_p \times D_p$ and *have full rank*. Hence, we expressed $\boldsymbol{A}$, $\boldsymbol{B}$ as some linear combination of the columns of $\boldsymbol{Q}$. Therefore

$$\begin{aligned} \boldsymbol{P}_A &= \boldsymbol{Q}\boldsymbol{G}_A(\boldsymbol{G}_A^{\mathrm{T}}\boldsymbol{G}_A)^{-1}\boldsymbol{G}_A^{\mathrm{T}}\boldsymbol{Q}^{\mathrm{T}}, \\ \boldsymbol{P}_B &= \boldsymbol{Q}\boldsymbol{G}_B(\boldsymbol{G}_B^{\mathrm{T}}\boldsymbol{G}_B)^{-1}\boldsymbol{G}_B^{\mathrm{T}}\boldsymbol{Q}^{\mathrm{T}}, \end{aligned} \tag{A.3}$$

and since $\boldsymbol{G}_A$, $\boldsymbol{G}_B$ are of full rank $\boldsymbol{G}_A(\boldsymbol{G}_A^{\mathrm{T}}\boldsymbol{G}_A)^{-1}\boldsymbol{G}_A^{\mathrm{T}} = \boldsymbol{G}_B(\boldsymbol{G}_B^{\mathrm{T}}\boldsymbol{G}_B)^{-1}\boldsymbol{G}_B^{\mathrm{T}} = \boldsymbol{I}$. Hence, $\boldsymbol{P}_A = \boldsymbol{P}_B = \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}$.

Appendix B

Second Appendix

Given a set of N feature vectors $\boldsymbol{x}_i$ stored column-wise in a matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ we wish to minimize the criterion

$$J(\boldsymbol{F}) = \sum_{i=1}^{N} ||\boldsymbol{x}_i - \boldsymbol{F}\boldsymbol{z}_i||^2 + N\mathrm{tr}(\boldsymbol{F}\boldsymbol{H}\boldsymbol{F}^{\mathrm{T}}), \tag{B.1}$$

where

$$\begin{aligned} \boldsymbol{z}_i &= (\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{x}_i, \\ \boldsymbol{H} &= \sigma^2(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} + \sigma^2\boldsymbol{I})^{-1}, \end{aligned} \tag{B.2}$$

$\sigma^2 \geq 0$, $\boldsymbol{F}$ is a $D \times D_p$ transformation matrix of full column rank D_p. Let decompose $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{Q}$ by Singular Value Decomposition (SVD) so that $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}} = \boldsymbol{I}$, and since $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$ is positive semi-definite $\boldsymbol{D} = [d_{ii}]$ is a diagonal matrix with $d_{ii} \geq 0$. For $\tilde{\boldsymbol{F}} = \boldsymbol{F}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{D}^{-1/2}$ we get $\tilde{\boldsymbol{F}}^{\mathrm{T}}\tilde{\boldsymbol{F}} = \boldsymbol{I}$, hence columns of $\tilde{\boldsymbol{F}}$ are orthonormal.

Now, the criterion (B.1) can be written as

$$J(\boldsymbol{F}) = N\mathrm{tr}(\boldsymbol{C}) - N\mathrm{tr}(\boldsymbol{K}_1\boldsymbol{C}_F - \boldsymbol{K}_2), \tag{B.3}$$

$$\boldsymbol{C} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}},\ \boldsymbol{C}_F = \frac{1}{N}\sum_{i=1}^{N}(\tilde{\boldsymbol{F}}^{\mathrm{T}}\boldsymbol{x}_i)(\tilde{\boldsymbol{F}}^{\mathrm{T}}\boldsymbol{x}_i)^{\mathrm{T}}, \tag{B.4}$$

$$\boldsymbol{K}_1 = \left[\frac{d_{ii}^2 + 2d_{ii}\sigma^2}{(d_{ii} + \sigma^2)^2}\right],\ \boldsymbol{K}_2 = \left[\frac{d_{ii}\sigma^2}{d_{ii} + \sigma^2}\right], \tag{B.5}$$

and since $(d_{ii} + \sigma^2)^2 = d_{ii}^2 + 2d_{ii}\sigma^2 + \sigma^4 \geq d_{ii}^2 + 2d_{ii}\sigma^2$, recall that $d_{ii} \geq 0, \sigma^2 \geq 0$, the matrix $\boldsymbol{K}_1$ is positive semi-definite, and additionally the diagonal entries of $\boldsymbol{K}_1$ are less than one and equal to one if and only if $\sigma^2 = 0$. Note that $\tilde{\boldsymbol{F}}\boldsymbol{y} = \tilde{\boldsymbol{F}}\tilde{\boldsymbol{F}}^{\mathrm{T}}\boldsymbol{x}_i$ is an orthogonal projection of $\boldsymbol{x}_i$ onto the subspace spanned by columns of $\boldsymbol{F}$, while $\boldsymbol{F}\boldsymbol{y}$ is not.

Proof: Firstly, let focus on the second term in (B.1)

$$\begin{aligned} \mathrm{tr}(\boldsymbol{F}\boldsymbol{H}\boldsymbol{F}^{\mathrm{T}}) &= \mathrm{tr}(\sigma^2(\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{Q} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{Q}) = \mathrm{tr}(\sigma^2\boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{D} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{Q}) = \\ &= \mathrm{tr}(\sigma^2(\boldsymbol{D} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{D}) = \mathrm{tr}(\boldsymbol{K}_2). \end{aligned} \tag{B.6}$$

Before the rearrangement of the first term in (B.1) note that

$$
\begin{aligned}
&(\boldsymbol{I} - \boldsymbol{F}(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{F}^{\mathrm{T}})^2 = \\
&= \boldsymbol{I} + \boldsymbol{F}\boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{D} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{D}(\boldsymbol{D} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{Q}\boldsymbol{F}^{\mathrm{T}} - 2\boldsymbol{F}\boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{D} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{Q}\boldsymbol{F}^{\mathrm{T}} = \\
&= \boldsymbol{I} - \boldsymbol{F}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{D}^{-\frac{1}{2}}\left[\boldsymbol{D}^{\frac{1}{2}}(-(\boldsymbol{D} + \sigma^2\boldsymbol{I})^{-2}\boldsymbol{D} + 2(\boldsymbol{D} + \sigma^2\boldsymbol{I})^{-1})\boldsymbol{D}^{\frac{1}{2}}\right]\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{Q}\boldsymbol{F}^{\mathrm{T}} = \\
&= \boldsymbol{I} - \tilde{\boldsymbol{F}}\boldsymbol{K}_1\tilde{\boldsymbol{F}}^{\mathrm{T}}.
\end{aligned}
$$

Back to the first term in (B.1), we get

$$
\begin{aligned}
&\sum_{i=1}^{N} ||\boldsymbol{x}_i - \boldsymbol{F}\boldsymbol{z}_i||^2 = \sum_{i=1}^{N} ||\boldsymbol{x}_i - \boldsymbol{F}(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{x}_i||^2 = \\
&= \sum_{i=1}^{N} \mathrm{tr}\left(\boldsymbol{x}_i(\boldsymbol{I} - \boldsymbol{F}(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{F}^{\mathrm{T}})^2\boldsymbol{x}_i^{\mathrm{T}}\right) = \mathrm{tr}(\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}) - \mathrm{tr}(\boldsymbol{K}_1\sum_{i=1}^{N}\tilde{\boldsymbol{F}}^{\mathrm{T}}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}\tilde{\boldsymbol{F}}) = \\
&= N\mathrm{tr}(\boldsymbol{C}) - N\mathrm{tr}(\boldsymbol{K}_1\boldsymbol{C}_F). \qquad \text{(B.7)}
\end{aligned}
$$

And finally, after combining (B.6) with (B.7) we obtain (B.3). ■

Appendix C

Third Appendix

Let $\boldsymbol{\Sigma}$ be a $D \times D$ square symmetric matrix of full-rank, let $\boldsymbol{G}$ be a $D \times D_r$ rectangular matrix and $D \geq D_r$, and let $\boldsymbol{\Psi}$ be a $D_r \times D_r$ square symmetric matrix of full-rank. Some useful matrix inverse identities are

$$\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\boldsymbol{G}(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{\Psi})^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1} = (\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{\Psi}^{-1}\boldsymbol{G}^{\mathrm{T}})^{-1}, \tag{C.1}$$

$$\boldsymbol{\Sigma}^{-1}\boldsymbol{G}(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{G} + \boldsymbol{\Psi})^{-1} = (\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{\Psi}^{-1}\boldsymbol{G}^{\mathrm{T}})^{-1}\boldsymbol{G}\boldsymbol{\Psi}^{-1}. \tag{C.2}$$

Combining these two identities we get

$$\boldsymbol{\Sigma}^{-1} - (\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{\Psi}^{-1}\boldsymbol{G}^{\mathrm{T}})^{-1}\boldsymbol{G}\boldsymbol{\Psi}^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1} = (\boldsymbol{\Sigma} + \boldsymbol{G}\boldsymbol{\Psi}^{-1}\boldsymbol{G}^{\mathrm{T}})^{-1}. \tag{C.3}$$

Appendix D

Fourth Appendix

The terminology concerning the speaker evaluation process can be summarized as[1]:

1. Test – a collection of trials constituting an evaluation component.
2. Trial – the individual evaluation unit involving a test segment and a hypothesized speaker.
3. Target (model) speaker – the hypothesized speaker of a test segment, one for whom a model has been created from training data.
4. Non-target (impostor) speaker – a hypothesized speaker of a test segment who is in fact not the actual speaker.
5. Segment speaker – the actual speaker in a test segment.
6. Target (true speaker) trial – a trial in which the actual speaker of the test segment is in fact the target (hypothesized) speaker of the test segment.
7. Non-target (impostor) trial – a trial in which the actual speaker of the test segment is in fact not the target (hypothesized) speaker of the test segment.
8. Turn – the interval in a conversation during which one participant speaks while the other remains silent.

[1] the terminology is taken from the glossary in http://www.itl.nist.gov/iad/mig/tests/spk/2008/sre08_evalplan_release4.pdf

Appendix E

Fifth Appendix

The Equal Error Rates (EERs) obtained for PLDA models trained on different development corpora (NIST040506, SW1, SW2 and SWC – for details on corpora see Section 7.1). The performance of PLDA based systems was tested on trials from NIST Speaker Recognition Evaluation (SRE) 2008 and on trials from NIST SRE 2010. EERs are depicted in dependence on the dimension D_h, D_w of latent variables $\boldsymbol{h}_i$, $\boldsymbol{w}_{ij}$ described in Section 6.2, respectively. Results are shown in Figure E.1 – Figure E.8. Graph in the first column is for females and in the second column for males. Above each graph the maximal, minimal and median value of EER are given along with dimension of latent variables for which they occur (given in brackets).

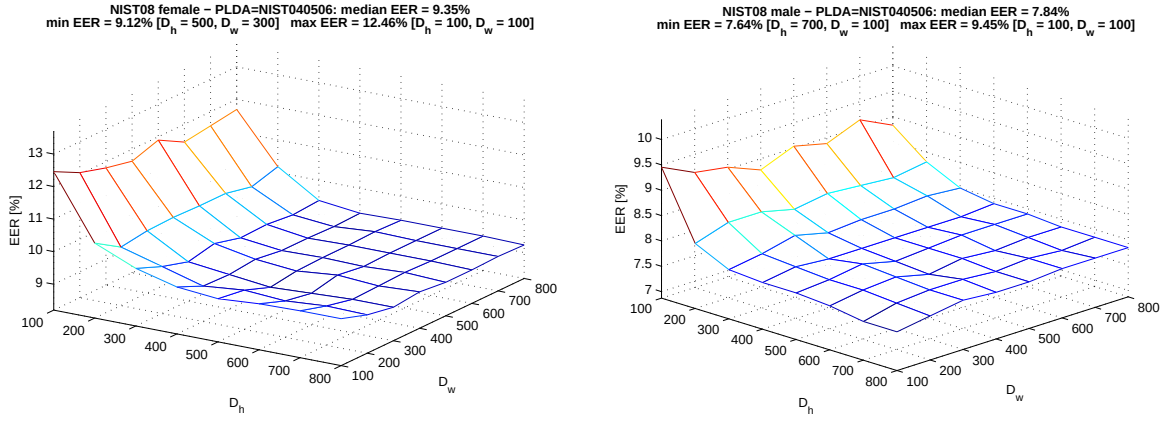

Figure E.1: One PLDA model trained on i-vectors extracted from corpus NIST040506 and tested on NIST08.

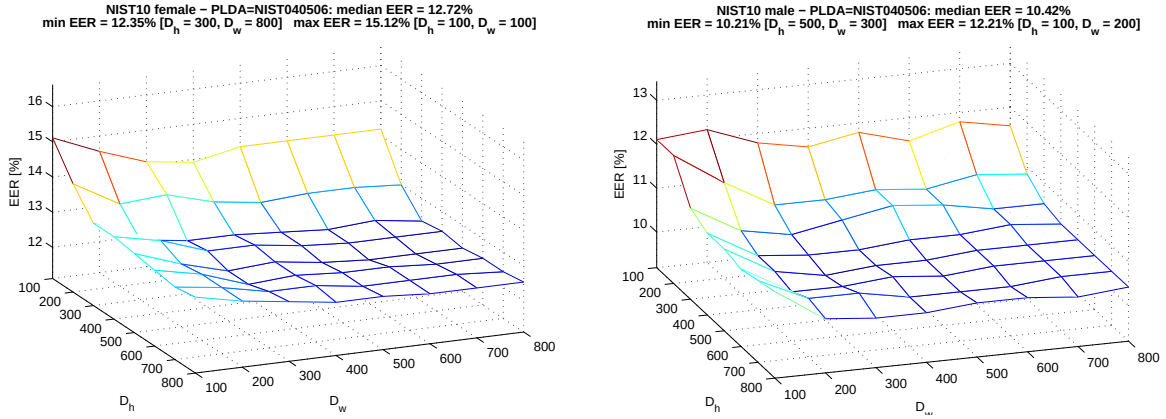

Figure E.2: One PLDA model trained on i-vectors extracted from corpus NIST040506 and tested on NIST10.

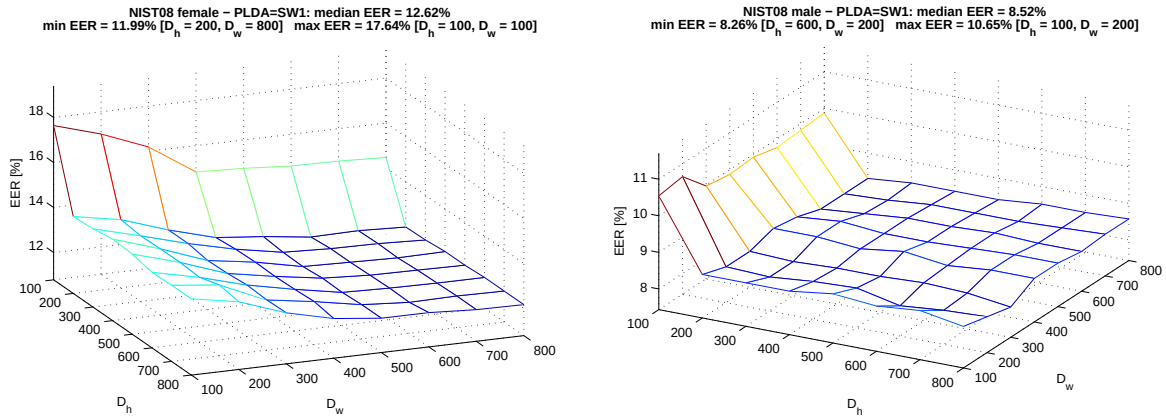

Figure E.3: One PLDA model trained on i-vectors extracted from corpus SW1 and tested on NIST08.

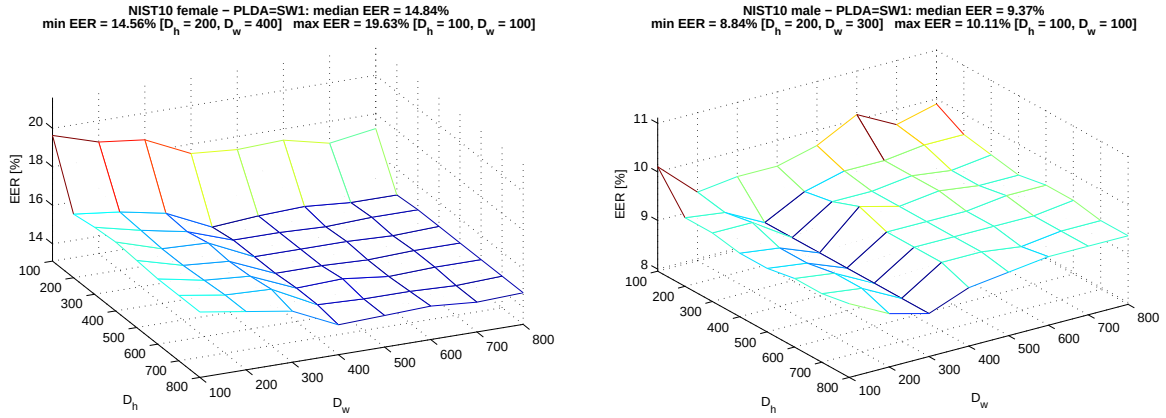

Figure E.4: One PLDA model trained on i-vectors extracted from corpus SW1 and tested on NIST10.

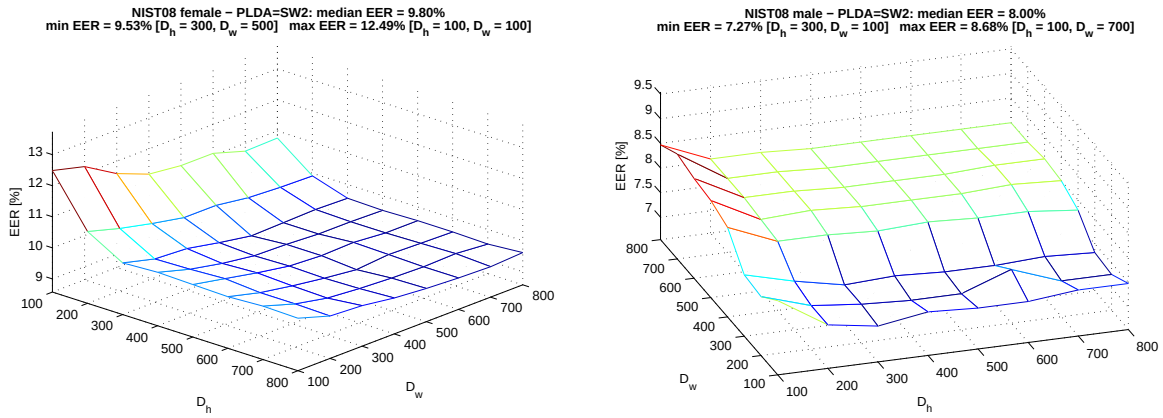

Figure E.5: One PLDA model trained on i-vectors extracted from corpus SW2 and tested on NIST08.

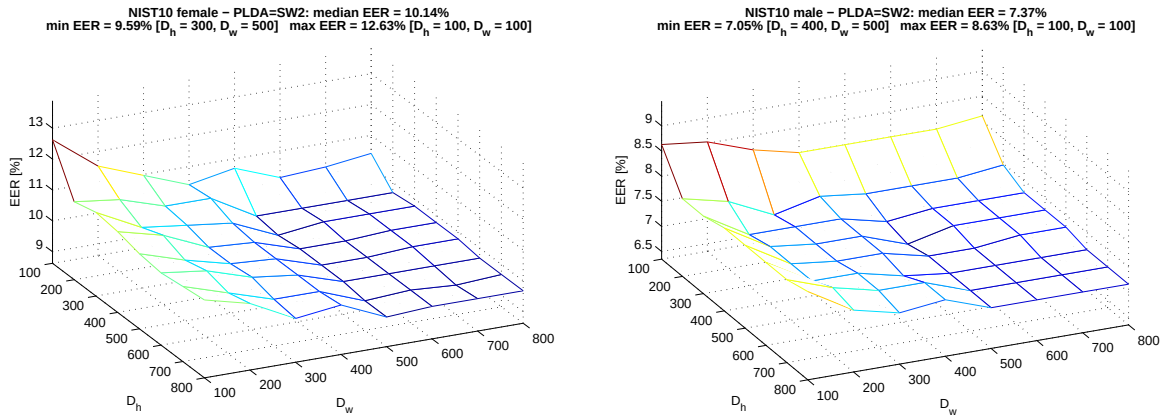

Figure E.6: One PLDA model trained on i-vectors extracted from corpus SW2 and tested on NIST10.

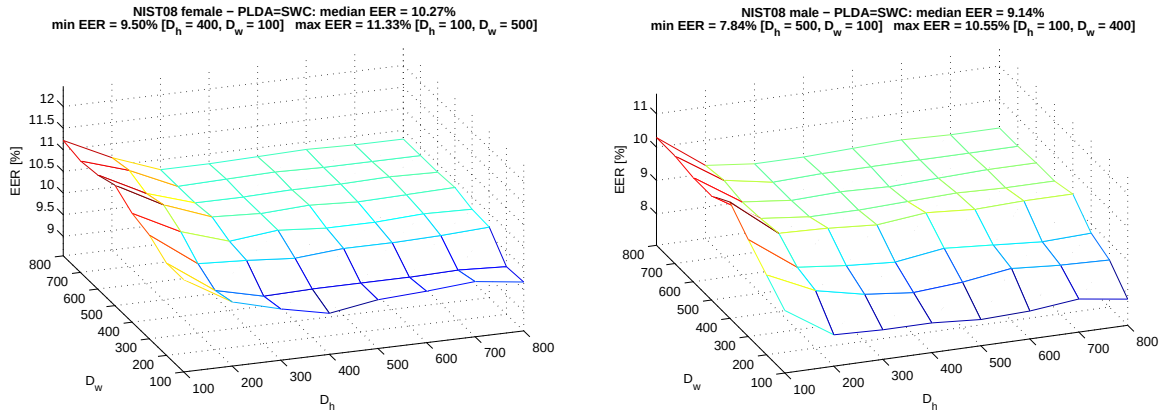

Figure E.7: One PLDA model trained on i-vectors extracted from corpus SWC and tested on NIST08.

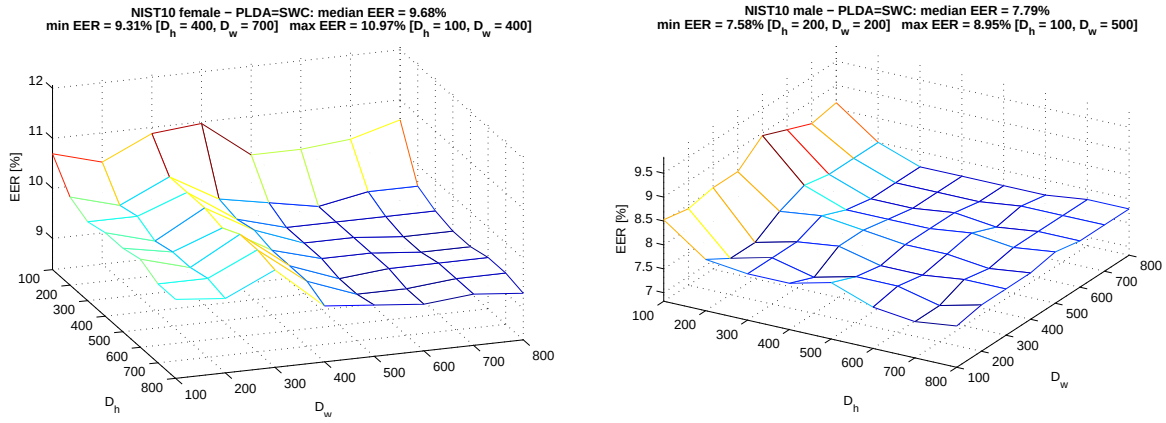

Figure E.8: One PLDA model trained on i-vectors extracted from corpus SWC and tested on NIST10.

Appendix F

Sixth Appendix

Examples of Feature Warping (FW) described in Section 5.3 for different types of datasets are shown in Figure F.1 – Figure F.6. First plot depicts the original dataset, in the second one data are rank normalized independently along each dimension, and in the third plot the inverse of the normal cumulative distribution function is computed for each dimension of each feature vector. For each plot the histograms in each dimension are drawn below and next to respective axis. Note that the shape of the dataset and the "direction/rotation" of the covariance of the dataset are preserved. The features in each dimension are pulled towards the zero. Moreover, also the variances in each local area of the feature space are partially preserved.

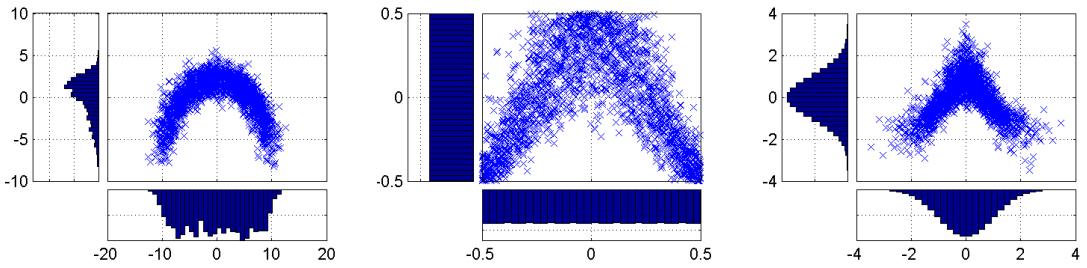

Figure F.1

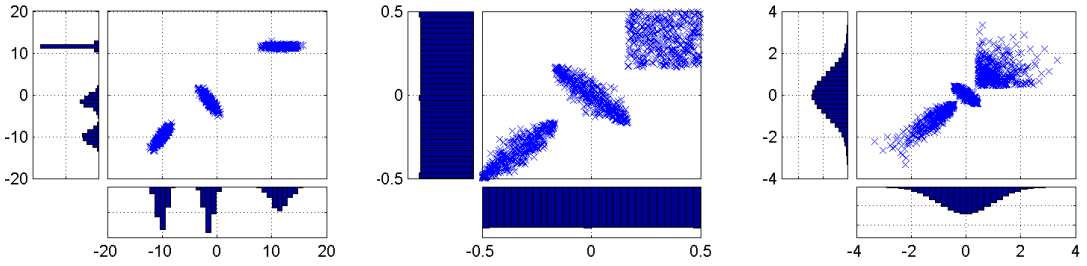

Figure F.2

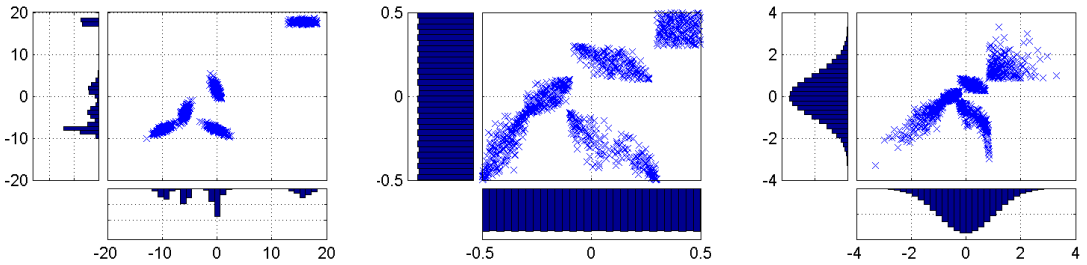

Figure F.3

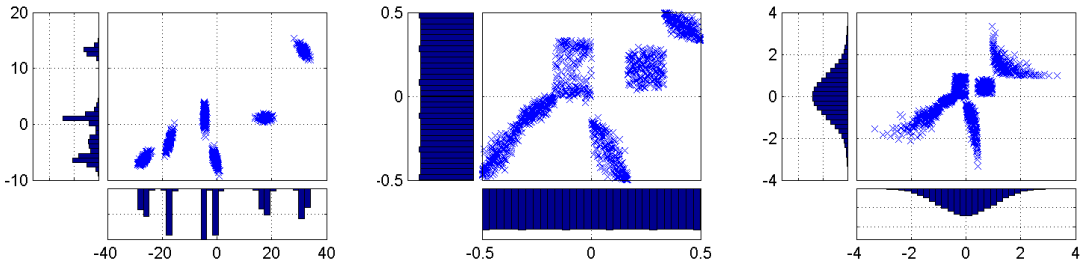

Figure F.4

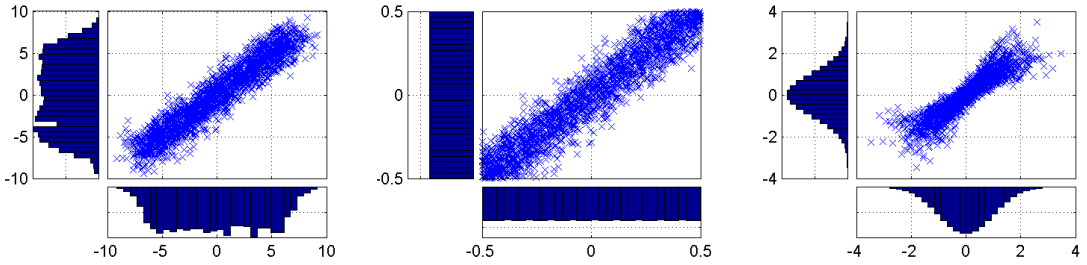

Figure F.5

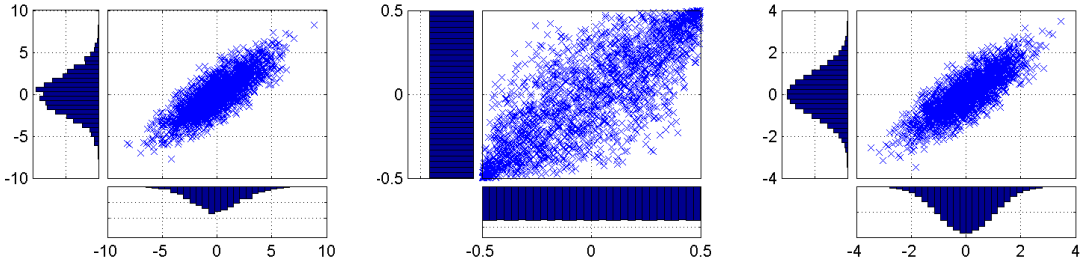

Figure F.6

Bibliography

[1] F. Bimbot, J. F. Bonastre, C. Fredouille, G. Gravier, M. I. Chagnolleau, S. Meignier, T. Merlin, O. J. Garcia, P. Delacretaz, and D. A. Reynolds, "A Tutorial on Text-Independent Speaker Verification," *EURASIP Journal on Applied Signal Processing*, vol. 4, pp. 430–451, 2004.

[2] H. Hermansky, "Perceptual Linear Predictive (PLP) Analysis of Speech," *The Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[3] J. Černocký, "TRAPS in All Senses, report of post-doc research internship," Tech. Rep., 2001.

[4] L. Quan and S. Bengio, "Hybrid Generative-Discriminative Models for Speech and Speaker Recognition," Dalle Molle Institute for Perceptual Artificial Intelligence, Martigny, Valais, Switzerland, Tech. Rep., 2002.

[5] N. Dehak, "Discriminative and Generative Approaches for Long- and Short-term Speaker Characteristics Modeling: Application to Speaker Verification," Ph.D. dissertation, École de Technologie Supérieure, Université du Québec, 2009.

[6] P. Matějka, O. Glembek, F. Castaldo, J. Alam, O. Plchot, P. Kenny, L. Burget, and J. Černocký, "Full-covariance UBM and Heavy-tailed PLDA in I-Vector Speaker Verification," *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011*, pp. 4828–4831, 2011.

[7] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A Performance Study of General-purpose Applications on Graphics Processors using CUDA," *Journal of Parallel and Distributed Computing*, vol. 68, no. 10, pp. 1370–1380, 2008.

[8] S. J. D. Prince and J. H. Elder, "Probabilistic Linear Discriminant Analysis for Inferences About Identity," *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8, 2007.

[9] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[10] N. D. Smith, "Using Augmented Statistical Models and Score Spaces for Classification," Ph.D. dissertation, University of Cambridge, 2003.

[11] D. A. Reynolds, "A Gaussian Mixture Modelling Approach to Text-independent Speaker Identification," Ph.D. dissertation, Georgia Institute of Technology, 1992.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[13] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[14] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Second Edition.* Elsevier Academic Press, 2003.

[15] X. Zhu, Y. Gao, S. Ran, F. Chen, I. Macleod, B. Millar, and M. Wagner, "Text-Independent Speaker Recognition Using VQ, Mixture Gaussian VQ and Ergodic HMMs," *ESCA Workshop on Automatic Speaker Recognition, Identification, and Verification*, pp. 55–58, 1994.

[16] V.Vapnik, *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.

[17] B. Schölkopf and A. J. Smola, *Learning with Kernels.* MIT Press, 2002.

[18] J. C. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[19] T. Joachims, "SVM light," 2002, available from: http://svmlight.joachims.org [last visited: 16.8.2012].

[20] R. Collobert, S. Bengio, and C. Williamson, "SVMTorch: Support vector machines for large-scale regression problems," *Journal of Machine Learning Research*, vol. 1, pp. 143–160, 2001, available from: http://bengio.abracadoudou.com/SVMTorch.html [last visited: 16.8.2012].

[21] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, available from: http://www.csie.ntu.edu.tw/~cjlin/libsvm [last visited: 16.8.2012].

[22] T. Hastie and R. Tibshirani, "Classification by Pairwise Coupling," *The Annals of Statistics*, pp. 507–513, 1998.

[23] K. Morik, P. Brockhausen, and T. Joachims, "Combining Statistical Learning with a Knowledge-based Approach - A case Study in Intensive Care Monitoring," *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-99)*, pp. 268–277, 1999.

[24] M. Katz, M. Schafföner, S. E. Krüger, and A. Wendemuth, "Score Calibrating for Speaker Recognition based on Support Vector Machines and Gaussian Mixture Models," *SIP '07, Proceedings of the Ninth IASTED International Conference on Signal and Image Processing*, pp. 139–144, 2007.

[25] J. C. Platt, "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods," *Advances in Large Margin Classifiers*, pp. 61–74, 1999.

[26] D. A. Reynolds, "Comparison of Background Normalization Methods for Text-Independent Speaker Verification," *EUROSPEECH-1997, 5th European Conference on Speech Communication and Technology*, vol. 2, pp. 963–966, 1997.

[27] J.-l. Gauvain and C.-h. Lee, "Maximum A Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains," *IEEE Transactions on Speech and Audio Processing*, vol. 2, pp. 291–298, 1994.

[28] A. Alexander, "Forensic Automatic Speaker Recognition Using Bayesian Interpretation and Statistical Compensation for Mismatched Conditions," Ph.D. dissertation, 2005.

[29] D. Povey and G. Saon, "Feature and Model Space Speaker Adaptation with Full Covariance Gaussians," *Interspeech*, 2006.

[30] M. J. F. Gales, "Maximum Likelihood Linear Transformation for HMM-based Speech Recognition," *Computer Speech and Language*, vol. 12, pp. 75–98, 1998.

[31] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book (for HTK Version 3.4)*. Cambridge University Engineering Department, 2008.

[32] M. J. F. Gales, "The Generation and Use of Regression Class Trees for MLLR Adaptation," Cambridge University Engineering Department, Tech. Rep., 1996.

[33] Z. Zajíc, L. Machlica, and L. Müller, "Refinement Approach for Adaptation Based on Combination of MAP and fMLLR," *Lecture Notes in Computer Science*, vol. Volume 572, pp. 274–281, 2009.

[34] Z. Zajíc, L. Machlica, and L. Müller, "Initialization of fMLLR with Sufficient Statistics from Similar Speakers," *Lecture Notes in Computer Science*, vol. 6836/2011, pp. 187–194, 2011.

[35] L. Yongxin, E. Hakan, G. Yuqing, and M. Etienne, "Incremental On-line Feature Space MLLR Adaptation for Telephony Speech Recognition," *Interspeech*, pp. 1417–1420, 2002.

[36] D. Povey and K. Yao, "A Basis Representation of Constrained MLLR Transforms for Robust Adaptation," *Computer Speech & Language*, vol. 26, pp. 35–51, 2011.

[37] T. S. Jaakkola and D. Haussler, "Exploiting Generative Models in Discriminative Classifiers," *In Advances in Neural Information Processing Systems 11*, pp. 487–493, 1999.

[38] S. Fine, J. Navrátil, and R. A. Gopinath, "A Hybrid GMM/SVM Approach to Speaker Identification," *IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP'01*, vol. 1, pp. 417–420, 2001.

[39] V. Wan and S. Renals, "Evaluation of Kernel Methods for Speaker Verification and Identification," *IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP'02*, vol. 1, pp. I–669– I–672, 2002.

[40] N. D. Smith and M. J. F. Gales, "Using SVMs to Classify Variable Length Speech Patterns," Cambridge University Engineering Department, Tech. Rep., 2002.

[41] C. Longworth and M. J. F. Gales, "Parametric and Derivative Kernels for Speaker Verification," *Interspeech*, pp. 310–313, 2007.

[42] W. M. Campbell, D. E. Sturim, D. A. Reynolds, and A. Solomonoff, "SVM Based Speaker Verification using a GMM Supervector Kernel and NAP Variability Compensation," *Acoustics, Speech and Signal Processing. ICASSP Proceedings*, vol. 1, 2006.

[43] W. M. Campbell, "Generalized Linear Discriminant Sequence Kernels for Speaker Recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing.*, vol. 1, 2002.

[44] K. Lee, C. You, T. Kinnunen, and D. Zhu, "Characterizing Speech Utterances for Speaker Verification with Sequence Kernel SVM," *Proceedings of Interspeech, Brisbane*, pp. 1397–1400, 2008.

[45] J. Louradour, K. Daoudi, and F. Bach, "SVM Speaker Verification using an Incomplete Cholesky Decomposition Sequence Kernel," *The Speaker and Language Recognition Workshop. IEEE Odyssey 2006*, pp. 1–5, 2006.

[46] J. Louradour, K. Daoudi, and F. Bach, "Feature Space Mahalanobis Sequence Kernels: Application to SVM Speaker Verification," *IEEE transactions on audio, speech, and language processing*, vol. 15, pp. 2465–2475, 2007.

[47] K. A. Lee, C. You, H. Li, and T. Kinnunen, "A GMM-based Probabilistic Sequence Kernel for Speaker Verification," *Proc. Interspeech 2007*, pp. 294–297, 2007.

[48] P. J. Moreno, P. P. Ho, and N. Vasconcelos, "A Kullback-Leibler Divergence based Kernel for SVM Classification in Multimedia Applications," in *In Advances in Neural Information Processing Systems 16*, 2003.

[49] N. Dehak and G. Chollet, "Support Vector GMMs for Speaker Verification," *The Speaker and Language Recognition Workshop. IEEE Odyssey*, pp. 1–4, 2006.

[50] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support Vector Machines using GMM Supervectors for Speaker Verification," *Signal Processing Letters, IEEE*, vol. 13, no. 5, pp. 308–311, 2006.

[51] W. Campbell, "Weighted Nuisance Attribute Projection," *IEEE Speaker Odyssey Workshop*, 2010.

[52] M. N. Do, "Fast Approximation of Kullback-Leibler Distance for Dependence Trees and Hidden Markov Models," *Signal Processing Letters, IEEE*, vol. 10, no. 4, pp. 115–118, 2003.

[53] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman, "MLLR Transforms as Features in Speaker Recognition," *Proc. Eurospeech, Lisbon*, pp. 2425–2428, 2005.

[54] Z. N. Karam and W. M. Campbell, "A New Kernel for SVM MLLR Based Speaker Recognition," *Proc. Interspeech 2007, Antwerp, Belgium*, pp. 290–293, 2007.

[55] Z. N. Karam and W. M. Campbell, "A Multi-class MLLR Kernel for SVM Speaker Recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP'08*, pp. 4117–4120, 2008.

[56] G. Upton and I. Cook, *A Dictionary of Statistics*. Oxford University Press, 2004.

[57] C. Longworth and M. J. F. Gales, "A Generalised Derivative Kernel for Speaker Verification," *Proceedings of Interspeech, Brisbane*, pp. 1385–1388, 2008.

[58] M. Katz, S. E. Krüger, M. Schafföner, E. Andelic, and A. Wendemuth, "Speaker Identification and Verification using Support Vector Machines and Sparse Kernel Logistic Regression," *Lecture Notes in Computer Science*, vol. 153/2006, pp. 176–184, 2006.

[59] A. Stuhlsatz, H. G. Meier, and A. Wendemuth, "Maximum Margin Classification on Convex Euclidean Metric Spaces," *Computer Recognition Systems 2*, pp. 216–223, 2008.

[60] M. Ferras, C. C. Leung, C. Barras, and J. L. Gauvain, "Constrained MLLR for Speaker Recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'07*, vol. 4, pp. 53–56, 2007.

[61] V. Wan and S. Renals, "Speaker Verification Using Sequence Discriminant Support Vector Machines," *IEEE Transactions on Speech and Audio Processing*, vol. 13, pp. 203–210, 2005.

[62] J. Pelecanos and S. Sridharan, "Feature Warping for Robust Speaker Verification," *A Speaker Odyssey - The Speaker Recognition Workshop*, 2001.

[63] L. Burget, P. Matějka, P. Schwarz, O. Glembek, and J. Černocký, "Analysis of feature extraction and channel compensation in GMM speaker recognition system," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, pp. 1979–1986, 2007.

[64] V. Wan and W. M. Campbell, "Support Vector Machines for Speaker Verification and Identification," *Proceedings of the 2000 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing X*, vol. 2, pp. 775–784, 2000.

[65] A. O. Hatch, S. Kajarekar, and A. Stolcke, "Within-class Covariance Normalization for SVM-based Speaker Recognition," *Proceedings of the 9th International Conference on Spoken Language Processing (ICSLP-Interspeech 2006)*, pp. 1471–1474, 2006.

[66] R. Auckenthaler, "Score Normalization for Text-Independent Speaker Verification Systems," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 42–54, 2000.

[67] A. Rosenberg, J. Delong, C. Lee, B. Juang, and F. Soong, "The Use of Cohort Normalized Scores for Speaker Recognition," *ICSLP*, pp. 599–602, 1992.

[68] D. E. Sturim and D. A. Reynolds, "Speaker Adaptive Cohort Selection for Tnorm in Text-Independent Speaker Verification," *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'05*, vol. 1, pp. 741–744, 2005.

[69] D. Colibro, C. Vair, F. Castaldo, E. Dalmasso, and P. Laface, "Speaker Recognition Using Channel Factors Feature Compensation," *EUSIPCO*, 2006.

[70] A. Solomonoff, C. Quillen, and W. Campbell, "Channel Compensation for SVM Speaker Recognition," *Odyssey'04*, pp. 219–226, 2004.

[71] A. Solomonoff, W. Campbell, and I. Boardman, "Advances in Channel Compensation for SVM Speaker Recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'05*, vol. 1, pp. 629–632, 2005.

[72] D. A. Harville, *Matrix Algebra From a Statistician's Perspective*. Springer, 2008.

[73] K. M. Abadir and J. R. Magnus, *Matrix Algebra (Econometric Exercises)*. Cambridge University Press, 2005.

[74] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Springer, 2007.

[75] B. Vesnicer and F. Mihelič, "The Likelihood Ratio Decision Criterion for Nuisance Attribute Projection in GMM Speaker Verification," *EURASIP Journal on Applied Signal Processing*, 2008.

[76] P. Kenny, "Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms," Centre de Recherche Informatique de Montréal (CRIM), Tech. Rep., 2006.

[77] P. Kenny and P. Dumouchel, "Disentangling Speaker and Channel Effects in Speaker Verification," *ICASSP*, vol. 1, pp. 37–40, 2004.

[78] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Factor Analysis Simplified," *ICASSP*, vol. 1, pp. 637–640, 2005.

[79] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*. Technical University of Denmark, 2008, available from: http://www2.imm.dtu.dk/pubdb/p.php?3274 [last visited: 16.8.2012].

[80] M. E. Tipping and C. M. Bishop, "Mixtures of Probabilistic Principal Component Analysers," *Neural Computation 11*, vol. 2, pp. 443–482, 1999.

[81] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of I-vector Length Normalization in Speaker Recognition Systems," *Interspeech*, pp. 249–252, 2011.

[82] P. Kenny and P. Dumouchel, "Experiments in Speaker Verification using Factor Analysis Likelihood Ratios," *IEEE Odyssey*, vol. 1, pp. 219–226, 2004.

[83] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint Factor Analysis Versus Eigenchannels in Speaker Recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, pp. 1435–1447, 2007.

[84] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel, "A Study of Interspeaker Variability in Speaker Verification," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, pp. 980–988, 2008.

[85] P. Kenny, N. Dehak, V. Gupta, P. Ouellet, and P. Dumouchel, "A New Training Regimen for Factor Analysis of Speaker Variability," 2008.

[86] O. Glembek, L. Burget, P. Kenny, M. Karafiát, and P. Matějka, "Simplification and Optimization of I-Vector Extraction," *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, pp. 4516–4519, 2011.

[87] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Speaker Adaptation Using an Eigenphone Basis," *IEEE Transactions on Speech and Audio Processing*, vol. 12, pp. 579–589, 2004.

[88] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Speaker and Session Variability in GMM-Based Speaker Verification," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, pp. 1448–1460, 2007.

[89] O. Glembek, L. Burget, P. Kenny, N. Dehak, and N. Brümmer, "Comparison of Scoring Methods used in Speaker Recognition with Joint Factor Analysis," *ICASSP*, vol. 1, pp. 4057–4060, 2009.

[90] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-End Factor Analysis For Speaker Verification," *IEEE Transactions on Audio, Speech and Language Processing*, 2010.

[91] P. Kenny, "Bayesian Speaker Verification with Heavy-Tailed Priors," *Keynote presentation, Odyssey Speaker and Language Recognition Workshop*, 2010.

[92] L. Machlica and Z. Zajíc, "Factor Analysis and Nuisance Attribute Projection Revisited," *Interspeech*, 2012.

[93] C. C. Joseph, J. P. Campbell, H. Nakasone, D. Miller, and K. Walker, "The Mixer Corpus of Multilingual, Multichannel Speaker Recognition Data," *Proc. 4th International Conference on Language Resources and Evaluation*, pp. 26 – 28, 2004.

[94] A. Martin, A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET Curve in Assessment of Detection Task Performance," vol. 4, pp. 1895 – 1898, 1997.

[95] N. Brümmer, "FoCal: Tools for Fusion and Calibration of Automatic Speaker Detection Systems," 2006, available from: http://sites.google.com/site/nikobrummer/focal [last visited: 16.8.2012].

[96] R. Dehak, N. Dehak, P. Dumouchel, and P. Kenny, "Linear and Non Linear Kernel GMM SuperVector Machines for Speaker Verification," *Interspeech*, vol. 1, pp. 302–305, 2007.

[97] N. Scheffer, Y. Lei, L. Ferrer, S. R. I. International, and M. Park, "Factor Analysis Back Ends for MLLR Transforms in Speaker Recognition," *Interspeech*, no. August, pp. 257–260, 2011.

[98] L. Machlica and Z. Zajíc, "Analysis of the Influence of Speech Corpora in the PLDA Verification," *Lecture Notes in Computer Science*, 2012.

[99] N. Dehak, R. Dehak, J. Glass, D. A. Reynolds, and P. Kenny, "Cosine Similarity Scoring without Score Normalization Techniques," *Proc. IEEE Odyssey Workshop, Brno, Czech Republic*, 2010.

[100] S. Marcel, C. McCool, P. Matějka, T. Ahonen, J. Černocký, S. Chakraborty, V. Balasubramanian, S. Panchanathan, C. H. Chan, J. Kittler, N. Poh, B. Fauve, O. Glembek, O. Plchot, Z. Jančík, A. Larcher, C. Lévy, D. Matrouf, J.-F. Bonastre, P.-H. Lee, J.-Y. Hung, S.-W. Wu, Y.-P. Hung, L. Machlica, J. Mason, S. Mau, C. Sanderson, D. Monzo, A. Albiol, H. V. Nguyen, L. Bai, Y. Wang, M. Niskanen, M. Turtinen, J. A. Nolazco-Flores, L. P. Garcia-Perera, R. Aceves-Lopez, M. Villegas, and R. Paredes, "On the results of the first mobile biometry (MOBIO) face and speaker verification evaluation," *ICPR'10 Proceedings of the 20th International conference on Recognizing patterns in signals, speech, images, and videos*, pp. 210–225, Aug. 2010.

[101] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 Algorithms in Data Mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2007.

[102] J. Vaněk, J. Trmal, and J. Psutka, "Optimization of the Gaussian Mixture Model Evaluation on GPU," *Interspeech*, pp. 1737–1740, 2011.

[103] A. D. Pangborn, *Scalable data clustering using GPUs*. Rochester Institute of Technology, Master Thesis, 2010.

[104] L. Machlica, J. Vaněk, and Z. Zajíc, "Fast Estimation of Gaussian Mixture Model Parameters on GPU using CUDA," *The 12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp. 167–172, 2011.

[105] C. Plant and C. Böhm, "Parallel EM-Clustering: Fast Convergence by Asynchronous Model Updates," *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pp. 178–185, 2010.

[106] "NVIDIA CUDA TM, NVIDIA CUDA C programming guide version 3.2, 11.9.2010."

[107] D. B. Kirk and W.-m. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010.

[108] V. Volkov and J. W. Demmel, "Benchmarking GPUs to Tune Dense Linear Algebra," pp. 31:1–31:11, 2008.

[109] N. Kumar, S. Satoor, and I. Buck, "Fast Parallel Expectation Maximization for Gaussian Mixture Models on GPUs Using CUDA," pp. 103–109, 2009.

[110] H. Aronowitz and O. Barkan, "New Developments in Joint Factor Analysis for Speaker Verification," *Interspeech*, pp. 129–132, 2011.

[111] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[112] M. Titsias and N. D. Lawrence, "Bayesian Gaussian Process Latent Variable Model," *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, pp. 844–851, 2010.

[113] J. Trmal, "Spatio-temporal structure of feature vectors in neural network adaptation," Ph.D. dissertation, University of West Bohemia, 2012.

[114] Z. Zajíc, L. Machlica, and L. Müller, "Bottleneck ANN : dealing with small amount of data in shift-MLLR adaptation," *IEEE 11th International Conference on Signal Processing (ICSP)*, 2012.

Authored and Co-authored Works

1. L. Machlica, Z. Zajíc, "Analysis of the Influence of Speech Corpora in the PLDA Verification," *Lecture Notes in Computer Science*, 2012.

2. L. Machlica, Z. Zajíc, "Factor Analysis and Nuisance Attribute Projection Revisited," *Interspeech*, 2012.

3. Z. Zajíc, L. Machlica, L. Müller, "Initialization of Adaptation by Sufficient Statistics Using Phonetic Tree," *IEEE 11th International Conference on Signal Processing (ICSP)*, 2012.

4. Z. Zajíc, L. Machlica, L. Müller, "Bottleneck ANN: dealing with small amount of data in shift-MLLR adaptation," *IEEE 11th International Conference on Signal Processing (ICSP)*, 2012.

5. L. Machlica, J. Vaněk, Z. Zajíc, "Fast Estimation of Gaussian Mixture Model Parameters on GPU using CUDA," *The 12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2011.

6. Z. Zajíc, L. Machlica, L. Müller, "Initialization of fMLLR with Sufficient Statistics from Similar Speakers," *Lecture Notes in Computer Science*, 2011.

7. L. Machlica, Z. Zajíc, L. Müller, "Discriminative adaptation based on fast combination of DMAP and DfMLLR," *Interspeech*, 2010.

8. L. Machlica, Z. Zajíc, L. Müller, "Robust Statistic Estimates for Adaptation in the Task of Speech Recognition," *Lecture Notes in Computer Science*, 2010.

9. L. Machlica, J. Vaněk, "UWB System Description for NIST SRE 2010," *NIST 2010 Speaker Recognition Evaluation Workshop – Odyssey's satellite*, 2010.

10. S. Marcel, C. McCool, P. Matějka, T. Ahonen, J. Černocký, S. Chakraborty, V. Balasubramanian, S. Panchanathan, C. H. Chan, J. Kittler, N. Poh, B. Fauve, O. Glembek, O. Plchot, Z. Jančík, A. Larcher, C. Lévy, D. Matrouf, J.-F. Bonastre, P.-H. Lee, J.-Y. Hung, S.-W. Wu, Y.-P. Hung, L. Machlica, J. Mason, S. Mau, C. Sanderson, D. Monzo, A. Albiol, H. V. Nguyen, L. Bai, Y. Wang, M. Niskanen, M. Turtinen, J. A. Nolazco-Flores, L. P. Garcia-Perera, R. Aceves-Lopez, M. Villegas, R. Paredes, "On the results of the first mobile biometry (MOBIO) face and speaker verification evaluation," *ICPR'10 Proceedings of the 20th International Conference on Recognizing Patterns in Signals, Speech, Images, and Videos*, 2010.

11. L. Machlica, J. Vaněk, "UWB system description: EVALITA 2009", *Conference of the Italian Association for Artificial Intelligence*, 2009.

12. Z. Zajíc, L. Machlica, L. Müller, "Refinement Approach for Adaptation Based on Combination of MAP and fMLLR," *Lecture Notes in Computer Science*, 2009.

13. A. Pražák, Z. Zajíc, L. Machlica, J. V. Psutka, "Fast Speaker Adaptation in Automatic Online Subtitling," *SIGMAP*, 2009.

14. L. Machlica, Z. Zajíc, A. Pražák, "Methods of Unsupervised Adaptation in Online Speech Recognition," *SPECOM Proceedings*, 2009.

15. Z. Zajíc, L. Machlica, A. Padrta, J. Vaněk, V. Radová, "An Expert System in Speaker Verification Task," *Interspeech*, 2008.

16. Z. Zajíc, J. Vaněk, L. Machlica, A. Padrta, "A Cohort Methods for Score Normalization in Speaker Verification System, Acceleration of On-line Cohort Methods," *SPECOM Proceedings*, 2007.

17. L. Machlica, Z. Zajíc, "The Speaker Adaptation of an Acoustic Model," *The 1st Young Researchers Conference on Applied Sciences*, 2007.