ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Faculty of Applied Sciences
Department of Cybernetics

Hybrid Modelling in Speaker Recognition

PhD THESIS REPORT

Ing. Lukáš Machlica

advisor: Doc. Dr. Ing. Vlasta Radová

Pilsen, 2008

Contents

List of Figures

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
EM	Expectation Maximization
FE	Feature Extraction
FI	Fisher Information
fMLLR	Feature Maximum Likelihood Linear Regression
FSW	Feature Space Whitening
GDK	Generalized Derivative Kernel
GLDS	Generalized Linear Discriminant Sequence
GMM	Gaussian Mixture Model
HLDA	Heteroscedastic Linear Discriminant Analysis
HMM	Hidden Markov Model
KKT	Karush-Kuhn-Tucker
KLD	Kulback-Leibler Divergence
L^2IPK	L^2 Inner Product Kernel
LDA	Linear Discriminant Analysis
LLRS	Log-Likelihood Ratio Score
LLS	Log-Likelihood Score
LPC	Linear Predictive Coding
LR	Logistic Regression

LVCSR	Large-Vocabulary Continuous Speech Recognition system
MAP	Maximum A-posteriory Probability
MFCC	Mel Frequence Cepstral Coefficients
ML	Maximum Likelihood
MLLR	Maximum Likelihood Linear Regression
MRF	Markov Random Field
NAP	Nuissance Attribute Projection
NN	Neural Networks
PCA	Principial Component Analysis
PDK	Probabilistic Distance Kernel
PLP	Perceptual Linear Prediction
PS	Probability Sequence
RBF	Radial Basis Function
RN	Rank Normalization
RT	Regression Tree
SD	Speaker Dependent
SI	Speaker Independent
SKLR	Sparse Kernel Logistic Regression
SLK	Supervector Linear Kernel
SN	Spherical Normalization
SR	Speaker Recognition
SVM	Support Vector Machine
T-norm	Test Normalization
TRAP	Temporal Pattern Features
UBM	Universal Background Model
VT	Vocal Tract
Z-norm	Zero Normalization
ZT-norm	Zero-Test Normalization

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. The math (especially the branch concerning numerical methods) has been spread and successfully applied to many scientific fields, 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 speech recognition the speaker 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 are requested (e.g. in telecommunications or in security domains).

The speaker recognition task 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 open set identification is considered, we assume that the unknown speaker stems from the set of reference speakers. In the open set identification such an assumption is broken. 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 or sibilants) present in the spoken speech, whereas the *text-independent* SR puts no limitation on what has been said.

The mathematic methods involved in SR are closely related to the human perception of sounds. Generally, the speech sound is a product of air expiration from lungs and vocal tract (VT) configuration. An important assumption concerning biometrics (e.g. the usage of SR in forensics) is that the VT changes between speakers. More precisely, each speaker's VT should be unique. Hence, speaker recognition task may be stated as the effort to capture the specific shapes of VT through investigation of the acoustic sound wave produced by the speaker.

During the last fifteen years an extensive progress has been done in the area of SR. The

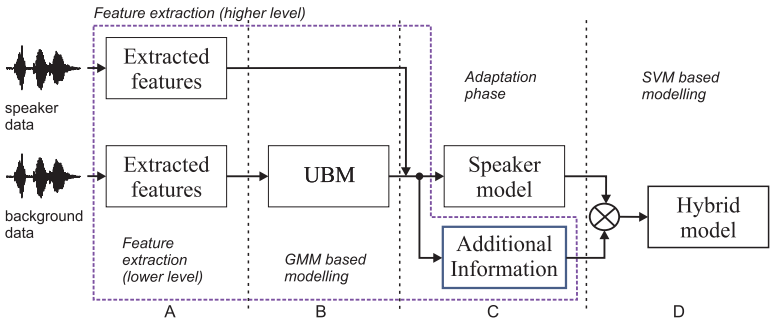

Figure 1.1: A: speaker data (spoken speech) and background data (acoustic sound waves representing the acoustic environment) are parametrized (features are extracted). B: The Universal Background Model is estimated according to the background data. C: The speaker specific feature vectors are utilized in the adaptation phase and a speaker model is constructed. Also an additional information about the shifting of UBM mixtures when fitting the speaker data is acquired. The procedure of such an additional information retrieval can be thought of as a higher level feature extraction and will be the aim of this report. D: Higher level features are used in the SVM training and finally a hybrid model is constructed. The speaker model from the phase C is involved in the SVM training only when some of the derivative mappings are used (for further details see Section 5.1.2).

problem of SR can be divided into several, separable tasks. Firstly, the varying time sequence of samples (amplitudes), forming the sound wave, has to be processed in a proper way to extract as many information regarding the speaker identity as possible. This is done in the feature extraction (FE) stage. The frequency response of particular parts of VT is given by the shape of VT. Hence, the feature extraction methods work primarily with the frequency spectrum of the sound signal. Some of these approaches will be mentioned in Chapter 2. The process of FE generates feature vectors, which are situated in the feature space – more precisely in speaker specific feature subspaces. In the second stage of SR, feature vectors are modeled and speaker models are obtained. Preferred methods exploit Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs) described in Chapter 3. Nowadays popular approaches of speaker modelling involve the adaptation techniques described in Chapter 4. Hence, an Universal Background Model (UBM) describing the acoustic environment is estimated, and the speaker specific models are adapted from the UBM. This is quite handy as the adaptation techniques allow that parts of the UBM that represent only the acoustic environment, remain unchanged for all the speakers. Moreover, we are able to track changes of each UBM mixture (assuming that GMMs are employed), in the adaptation process. Loosely speaking, *we are able to determine, in which direction and how far has been each of the mixtures moved to fit the speaker data.* The information about the shifting of UBM mixtures can be seen as a new advice that can be utilized in order to better capture the relations in the feature space. This is the main motivation of this report. It has turned out that the Support Vector Machine (SVM – introduced in Chapter 3), is a suitable tool to model such dependencies (see Chapter 6). Hence, in the training phase, both the generative and the discriminative model – *hybrid modelling* – are involved (see Figure 1.1). Appropriate methods will be introduced in Chapter 5. In the last phase of SR, an speaker model and parametrized utterance of an unknown speaker are given. The parametrized utterance is confronted with the reference speaker model so that a recognition score is obtained and further handled according to the stated task (verification/identification).

Chapter 2

Feature Extraction

Without a question, the feature extraction process can be regarded as the first and the most important step for the further manipulation with the data. The task can be divided into two subproblems, whether we want to classify the data or represent the data. In the first case 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 inter-class variation would be as high as possible and the intra-class variation would be as low as possible. In the latter case, we want to extract features that capture the true nature of the data. Both processes assume and should also reflect the understanding of behavior, structure, correctness of the data, etc. Complications like channel distortion, 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, distribution of samples) to facilitate the problem solving procedure. 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.

Let us focus now on the specific problem of the feature extraction in the task of speaker recognition, more precisely in hybrid modelling. A plenty of methods were already developed and successfully tested to fulfill the task's requirements. As will be shown soon, the specific task of hybrid modelling involves two feature extraction stages. In the first stage, features corresponding to acoustic events are extracted directly from the acoustic sound wave. Well known methods used for this purpose are Linear Predictive Coding (LPC), Mell Frequency Cepstral Coefficients (MFCC) [BBF+04], Perceptual Linear Prediction (PLP) [Her90], Temporal Pattern (TRAP) features [Čer01] and many others. In the second stage, after some manipulation with the data and retrieval of a suitable description of the data, the feature extraction process is applied again – higher level feature extraction (see Figure 1.1). Preferred methods are supervector (Section 5.1.1) and Fisher feature extraction techniques (Section 5.1.2). The supervector extraction technique consists of concatenation of important parameters related to the generative model, which subsequently form a high-dimensional vector - *supervector*. The Fisher feature techniques (and its generalizations) are based on the Fisher information and Fisher score retrieval. They are able to measure the information about the

suitability of the model imposed on the data structure, thus the deviation of the data from a model (see (5.43) and (5.44)).

The generative approach plays a role in the first part, whereas the discriminative training appears in the second part (in this work in the form of SVM training) of the extraction process. Hence, first of all a feature space is constructed. The particular speaker utterances are parametrized utilizing one of the before mentioned methods, thus a set of speaker dependent feature vectors is formed and spread over the feature space. Most often the Maximum A - Posteriori Probability adaptation (MAP - see Section 4.1) is performed to acquire an speaker model. Such a model can be thought of as a new source of additional informations, e.g. of metric properties in the newly formed space. Hence, we revealed a new informational supply and are ready to make use of it. Detailed analysis and procedures will be given in Chapter 5. An example of the described feature extraction process is depicted in Figure 2.1.

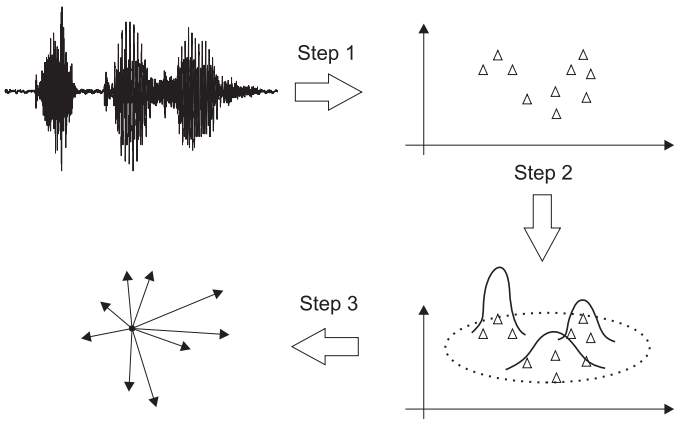

Figure 2.1: Feature extraction process involved in the task of hybrid modelling. Step 1: a feature vector is extracted directly from the acoustic sound wave employing one of the appropriate methods mentioned in the text. Step 2: retrieval of a convenient (e.g. statistical) description of the data. Step 3: features are mapped into a high-dimensional space, hence a supervector will be obtained.

Chapter 3

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 indeed, because each speaker 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) $D(\boldsymbol{x})$ 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 data corresponds to one of the given classes (e.g. each speakers' data were seen during the training). The classifier can be expressed as a mapping in the form

$$D : \boldsymbol{X} \mapsto \boldsymbol{Y}, \quad y_q = D(\boldsymbol{x}) \,, \tag{3.1}$$

where $\boldsymbol{Y} = \{y_1, \ldots, y_Q\}$ is the finite set of Q reference labels and $\boldsymbol{X}$ represents an open set of feature vectors spread in the feature space according to the 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).

3.1 Optimal Classifier

From the theoretical point of view an optimal classifier should minimize the *overall risk* R given as [DHS00]

$$R(D) = \int R(D(\boldsymbol{x})|\boldsymbol{x}) P(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \tag{3.2}$$

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

$$R(D(\boldsymbol{x})|\boldsymbol{x}) = \sum_{q=1}^{Q} l(D(\boldsymbol{x})|y_q)p(y_q|\boldsymbol{x}) \,, \tag{3.3}$$

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 as $i-th$ class. The loss can be regarded also as a cost function $c(\boldsymbol{x}, y_s, D(\boldsymbol{x}))$ for $D(\boldsymbol{x}) = y_i$, penalizing misclassifications so that the following holds

$$c(\cdot,\cdot,\cdot) \geq 0 \text{ and } c(\cdot, y_s, y_s) = 0 \,. \tag{3.4}$$

Now, the optimal classifier D^* should be chosen in order to minimize the following form

$$R(D) = \sum_{q=1}^{Q} \int_{\boldsymbol{X}} c(\boldsymbol{x}, y_q, D(\boldsymbol{x}))P(\boldsymbol{x}, y_q)\mathrm{d}\boldsymbol{x} \,. \tag{3.5}$$

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

$$R(D) = \int_{\boldsymbol{X}\times\boldsymbol{Y}} c(\boldsymbol{x}, y, D(\boldsymbol{x}))\mathrm{d}P(\boldsymbol{x}, y) \,. \tag{3.6}$$

Thus, all possible pairs of feature vectors and labels are considered and rated while (3.4) holds. One of the popular cost functions is the zero-one loss function defined as

$$c(\boldsymbol{x}, y_s, D(\boldsymbol{x})) = l(D(\boldsymbol{x})|y_s) = \begin{cases} 0 \text{ if } D(\boldsymbol{x}) = y_s \\ 1 \text{ if } D(\boldsymbol{x}) \neq y_s \end{cases} \,, \tag{3.7}$$

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

$$R(D(\boldsymbol{x})|\boldsymbol{x}) = \sum_{q=1}^{Q} p(y_q|\boldsymbol{x}) - p(y_s|\boldsymbol{x}) = 1 - p(y_s|\boldsymbol{x}) \,. \tag{3.8}$$

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

$$D^*(\boldsymbol{x}) = \underset{y_q \in \boldsymbol{Y}}{\arg\max}\{p(y_q|\boldsymbol{x})\}, \;\; \forall \boldsymbol{x} \in \boldsymbol{X} \,. \tag{3.9}$$

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 [Smi03]. An example of an optimal decision rule when three classes are present is shown in Figure 3.1.

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 (e.g. how well do they represent their parent class). The approaches will be presented in the sequel.

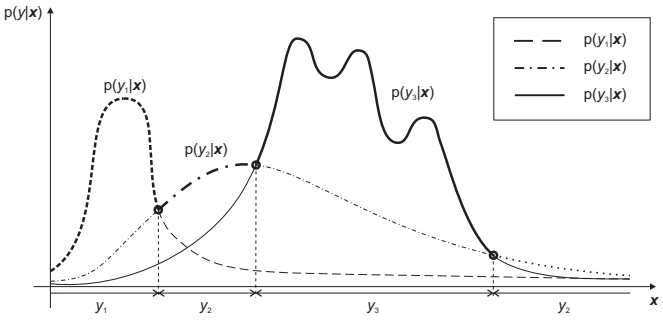

Figure 3.1: The a-posteriori probability distributions for classes y_1, y_2 and y_3 given the feature vector $\boldsymbol{x}$. The bold line represents the optimal decision rule that should the optimal classifier obey.

3.2 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 (3.9) indirectly utilizing the Bayes theorem, where the posterior probability for class y_q can be expressed as

$$p(y_q|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y_q)P(y_q)}{P(\boldsymbol{x})}, \tag{3.10}$$

and

$$P(\boldsymbol{x}) = \sum_{q=1}^{Q} p(\boldsymbol{x}|y_q)P(y_q) \tag{3.11}$$

is called the *evidence*. The decision rule (3.9) can be now rewritten into the form

$$D^*(\boldsymbol{x}) = \underset{y_q \in \boldsymbol{Y}}{\arg\max} \ \{\ln p(\boldsymbol{x}|y_q) + \ln P(y_q)\} \tag{3.12}$$

and for equiprobable classes

$$D^*(\boldsymbol{x}) = \underset{y_q \in \boldsymbol{Y}}{\arg\max} \ \{\ln p(\boldsymbol{x}|y_q)\} \ . \tag{3.13}$$

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 (3.11). We will focus on Gaussian Mixture Models (GMMs) and also the concept of Hidden Markov Models (HMMs) will be discussed.

3.2.1 Gaussian Mixture Model (GMM)

Gaussian Mixture Models were firstly introduced to the speaker recognition by Reynolds [Rey92] and are widely used up to now. For an I dimensional feature vector $\boldsymbol{x}$ the GMM

takes the form

$$g(\boldsymbol{x}) = p(\boldsymbol{x}|\boldsymbol{\lambda}) = \sum_{m=1}^{M} \omega_m p(\boldsymbol{x}|m, \boldsymbol{\lambda}) \,, \tag{3.14}$$

$$p(\boldsymbol{x}|m, \boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{C}_m) = \frac{1}{(2\pi)^{I/2} \left|\boldsymbol{C}_m\right|^{1/2}} \exp\left\{-0.5 \left(\boldsymbol{x} - \boldsymbol{\mu}_m\right)^{\mathrm{T}} \boldsymbol{C}_m^{-1} \left(\boldsymbol{x} - \boldsymbol{\mu}_m\right)\right\} , \tag{3.15}$$

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

$$\boldsymbol{\lambda} = \{\omega_m, \boldsymbol{\mu}_m, \boldsymbol{C}_m\}_{m=1}^{M} \,. \tag{3.16}$$

There are some restrictions laid on the mixture weights. These can be expressed as

$$\forall m : 0 \leq \omega_m \leq 1 \text{ and } \sum_{m=1}^{M} \omega_m = 1 \,. \tag{3.17}$$

Generally, the covariance matrix $\boldsymbol{C}_m$ is considered full, nevertheless in praxis mostly diagonal matrices are assumed (especially because of numerical stability and computational costs). It is obvious that after the class models have been trained, the label y_q in (3.12) or (3.13) is replaced by speaker specific model parameters $\boldsymbol{\lambda}_q$ and the classifier is brought to bear. Methods that consider also the prior probabilities $P(\boldsymbol{\lambda}_q)$ – prior information about the distribution of model parameters – are denoted as adaptation techniques and are discussed in Section 4. 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 [DLR77]. It is based on the Maximum Likelihood (ML) approach and tries to maximize the output probability of the model for submitted training data.

3.2.2 Hidden Markov Model (HMM)

Hidden Markov Models were developed in the 1960's by Baum and his colleagues [Rab89] 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 [TK03]. 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).

The Hidden Markov Model is characterized by a set of parameters $\boldsymbol{\Lambda} = \{\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{G}, \boldsymbol{\pi}\}$, where $\boldsymbol{S} = \{s_1, \ldots, s_J\}$ represents the set of J states (classes) 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

$$a_{ij} = p(s(t+1) = s_j | s(t) = s_i), \;\; i, j \in \{1, \ldots, J\} \, . \tag{3.18}$$

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

$$\pi_i = P(s(1) = s_i) \, . \tag{3.19}$$

Sometimes even final state probabilities are defined. And at last, the set $\boldsymbol{G} = \{\boldsymbol{\lambda}_{s_1}, \ldots, \boldsymbol{\lambda}_{s_J}\}$ defines the GMM parameters (see (3.16)) 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 path through the HMM and $\boldsymbol{\Psi}$ the entire set of such paths [Smi03], then

$$\begin{aligned} p(\boldsymbol{X}|\boldsymbol{\Lambda}) &= \sum_{\psi \in \boldsymbol{\Psi}} p(\boldsymbol{X}, \psi | \boldsymbol{\Lambda}) = \sum_{\psi \in \boldsymbol{\Psi}} p(\boldsymbol{X}|\psi, \boldsymbol{\Lambda}) p(\psi|\boldsymbol{\Lambda}) = \\ &= \sum_{\psi \in \boldsymbol{\Psi}} \left(\prod_{k=1}^{N} g^{\psi}_{s(k)}(\boldsymbol{x}_k) \right) \left(\prod_{k=1}^{N} a^{\psi}_{s(k),s(k+1)} \right) \\ &= \sum_{\psi \in \boldsymbol{\Psi}} \left(\prod_{k=1}^{N} g^{\psi}_{s(k)}(\boldsymbol{x}_k) \; a^{\psi}_{s(k),s(k+1)} \right) , \end{aligned} \tag{3.20}$$

where $g^{\psi}_{s(k)}(\boldsymbol{x}_k)$ is the state output probability defined in (3.14) and the upper index $^{\psi}$ indicates the presence in the state-level path ψ. Using directly (3.20) is computationally unbearable, therefore several efficient methods were developed, e.g. *forward-backward* or *Viterbi* algorithm (for details see [TK03]). Similarly to the GMM case, the classification is done according to

$$\boldsymbol{\Lambda}^* = \underset{\boldsymbol{\Lambda} \in L(\boldsymbol{\Lambda})}{\arg\max} \{p(\boldsymbol{X}|\boldsymbol{\Lambda})\} \, , \tag{3.21}$$

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

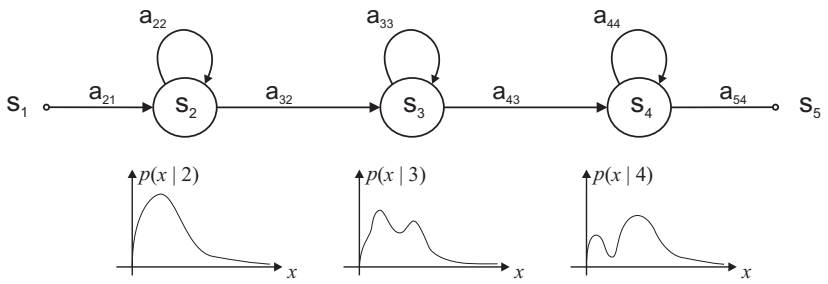

Figure 3.2: An example of a left-to-right, five state Hidden Markov Model with two non-emitting states s_1 and s_5. The transition probabilities are described by a_{ij} and the output probabilities of states s_2, s_3 and s_4 are represented by the GMMs (illustrated below each state).

ergodic models [ZGR+94]. 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 3.2.

3.3 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 (3.11) is not available, nonparametric classifiers cannot generate samples of the data. We will focus on linear discriminant functions in the form of

$$\mathcal{H} : f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b, \tag{3.22}$$

where $\boldsymbol{w}$ is the normal vector of the hyperplane $\mathcal{H}$ and the scalar b denotes the offset. 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 (3.22) does not define the geometrical distance of the point $\boldsymbol{x}$ from the hyperplane $\mathcal{H}$, this can be acquired according to

$$d(\boldsymbol{x}, \mathcal{H}) = \frac{f(\boldsymbol{x})}{\|\boldsymbol{w}\|} \,. \tag{3.23}$$

The classification obeys the rule

$$D(\boldsymbol{x}) = \begin{cases} y_1 \text{ if } f(\boldsymbol{x}) \geq 0 \\ y_2 \text{ if } f(\boldsymbol{x}) < 0 \end{cases} \,. \tag{3.24}$$

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), which will be now our main domain of interest.

3.3.1 Support Vector Machine (SVM)

Support Vector Machine was firstly introduced by Vapnik [V.V95]. 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 (3.24) in the manner of (3.25).

$$\begin{array}{l} \mathcal{H}_1 : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \geq +1 \text{ for } y_i = +1 \\ \mathcal{H}_2 : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \leq -1 \text{ for } y_i = -1 \end{array} \implies \forall i : y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1 \geq 0\,. \tag{3.25}$$

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 (3.23). 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 (3.25), Vapnik introduced positive variables ξ_i denoted also *slack variables*. The constraints become now

$$\begin{aligned} \mathcal{H}_1 &: \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \geq +1 - \xi_i \text{ for } y_i = +1\,, \\ \mathcal{H}_2 &: \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \leq -1 + \xi_i \text{ for } y_i = -1\,, \\ &\xi_i \geq 0\,. \end{aligned} \tag{3.26}$$

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 $\sum_i \xi_i$ is nothing else as the cost (loss) defined in (3.3). Now, the problem can be formulated as

$$\text{minimize} \quad \left[\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i\right]\,, \tag{3.27}$$

subject to (3.26), 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 [SS02]). An example of the SVM composition is depicted in Figure 3.3. Note that the square of $\boldsymbol{w}$ 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 [Bur98]). 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

$$L_{\mathrm{P}} = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\{y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1 + \xi_i\} - \sum_i \beta_i\xi_i\,. \tag{3.28}$$

Solving the problem (3.28) (seeking for mimimum) results in

$$\begin{aligned} \frac{\partial L_{\mathrm{P}}}{\partial \boldsymbol{w}} &= \boldsymbol{w} - \sum_i \alpha_i y_i \boldsymbol{x}_i &&= 0\,, \\ \frac{\partial L_{\mathrm{P}}}{\partial b} &= -\sum_i \alpha_i y_i &&= 0\,, \\ \frac{\partial L_{\mathrm{P}}}{\partial \xi_i} &= C - \alpha_i - \beta_i &&= 0\,, \end{aligned} \tag{3.29}$$

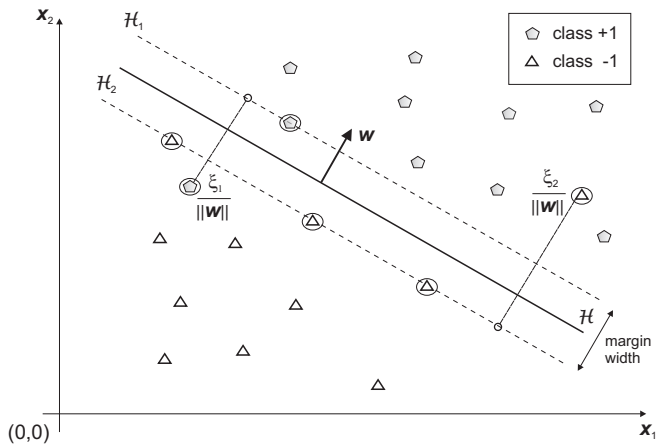

Figure 3.3: An example of the non-separable case of the SVM problem. Vectors encapsulated in circles denote support vectors.

with additional KKT conditions

$$\begin{aligned} y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1 + \xi_i &\geq 0\,, \\ \alpha_i\{y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1 + \xi_i\} &= 0\,, \\ \beta_i\xi_i &= 0\,, \\ \xi_i, \alpha_i, \beta_i &\geq 0\,. \end{aligned} \tag{3.30}$$

Substituting (3.29) to (3.28) gives us the dual formulation, which has to be *maximized*, in the form of

$$L_{\mathrm{D}} = \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j \tag{3.31}$$

subject to (3.30). According to (3.29) the normal vector $\boldsymbol{w}$ can be computed as

$$\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i\,. \tag{3.32}$$

Thus, the decision hyperplane (3.22) results in

$$\mathcal{H} : f(\boldsymbol{x}) = \sum_i \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x} + b\,. \tag{3.33}$$

After inspection of the KKT conditions it is quite clear that $\alpha_i \in (0, C)$ (consider the third condition in (3.29) and fourth condition in (3.30)). Furthermore, the second condition in (3.30) 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 (3.26)). Vectors with nonzero α_i are called the *support vectors* as just they participate in (3.32). The offset b cannot be computed directly – it does not occur in (3.29). However, it can be computed utilizing the second KKT condition in (3.30) choosing any i for which $\alpha_i \neq 0$ – it is numerically safer to take the mean value of b resulting from all such equations [Bur98].

Note that equations (3.31) and (3.33) depend on data 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, the dot products in (3.31) and (3.33) may be replaced by dot products defined in the space $\boldsymbol{\mathcal{X}}$ and 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* $K(\cdot, \cdot)$. Hence, the explicit information about the form of Φ is *unnecessary*. Now, the two equations (3.31), (3.33) can be rewritten as

$$L_{\mathrm{D}} = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \,, \tag{3.34}$$

$$\mathcal{H} : f(\boldsymbol{x}) = \sum_i \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b \,. \tag{3.35}$$

It should be also stated that the matrix

$$\boldsymbol{H} = [H_{ij}], \;\; H_{ij} = \frac{\partial^2 L_{\mathrm{D}}}{\partial \alpha_i \partial \alpha_j} = -y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{3.36}$$

represents the Hessian matrix, which is used to determine the global maximum of the optimization problem (3.34).

Some of the most familiar 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 – $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \left| \boldsymbol{x}_i - \boldsymbol{x}_j \right|^2)$.

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 5). Generally, any function representing a dot product in some space may be considered as kernel function. The condition that a function has to satisfy to be a valid kernel is known as *Mercer's condition* [V.V95]. Suppose any square integrable function $f(\boldsymbol{x})$,

$$\int f(\boldsymbol{x})^2 \mathrm{d}\boldsymbol{x} < \infty \,. \tag{3.37}$$

Then $K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \Phi_i(\boldsymbol{x})\Phi_i(\boldsymbol{y})$ is a valid kernel if and only if

$$\int K(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{x}) f(\boldsymbol{y}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y} \geq 0 \,. \tag{3.38}$$

Hence, the main aspects of the Mercer's condition are that the kernel function has to be symmetric ($K(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{y}, \boldsymbol{x})$) 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$^{\mathrm{light}}$ [Joa02] suitable for sparse data problems (e.g. derivative kernels – see Section 5.1.2) or also very popular SVMtorch [CBW01] and LibSVM [CL01].

3.4 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 mixtures. The situation is illustrated in Figure 3.4. 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 mixtures are employed, the mixtures will unnecessarily oppress each other. Such a problem is known as *overtraining* and decreases the ability of the classifier to generalize to unseen data. The proper choice of the number of GMM mixtures is task dependent, relevant factors are dimensionality and amount of training data.

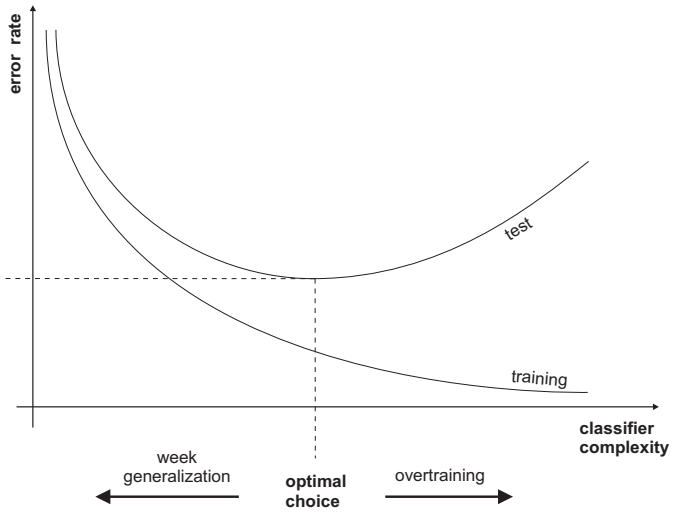

Figure 3.4: Overtraining phenomena.

Regarding the time consumption of parametric and nonparametric classifiers, it can be anticipated that nonparametric classifiers need longer time to be trained as they use all the training data at once to learn the decision rules. However, when comparing a complex generative model like HMM and discriminative classifier like SVM, the time consumption to train the SVM should be lesser. The data has to be seen only once, only the information about their dot products is stored and through the time many vectors will be neglected (only support vectors will left), whereas the training of HMM is an iterative one, hence it has to be run several times. Thus, the complexity plays a major role in the question of time consumption.

In this chapter only three classifiers, important for the successive explanation of the report objectives, were presented. Some other successfully exploited parametric classifiers are Markov Random Fields (MRFs) or Bayesian Networks (BNs), and in the case of non-parametric classifiers popular Neural Networks (NN) and Logistic Regression (LR) should be mentioned (for details see e.g. [TK03]).

Chapter 4

Adaptation Techniques

The main difference between the adaptation and ordinary training methods consists in the additional, prior knowledge about the distribution of model parameters, usually derived from the speaker independent (SI) model [PMMR07]. The adaptation adjusts the model so that the probability of the adaptation data would be maximized. This is equivalent to

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} p(\boldsymbol{O}^1, \ldots, \boldsymbol{O}^E|\boldsymbol{\lambda})p(\boldsymbol{\lambda}), \tag{4.1}$$

where $p(\boldsymbol{\lambda})$ stands for the prior information about the distribution of model parameters $\boldsymbol{\lambda}$ (in the case of GMM these are the weights, means and covariance matrices of particular mixtures), $\boldsymbol{O}^i = \{\boldsymbol{o}_1^i, \boldsymbol{o}_2^i, \ldots, \boldsymbol{o}_T^i\}, i = 1, \ldots, E$, is the sequence of feature vectors related to the $i-th$ speaker, $\boldsymbol{\lambda}^*$ is the best estimation of speaker dependent (SD) model parameters.

The function that will be optimized in the consequent text can be derived from the maximum likelihood approach and can be expressed as [PMMR07]

$$Q(\boldsymbol{\lambda}, \bar{\boldsymbol{\lambda}}) = \sum_{m=1}^{M} \sum_{t=1}^{T} p(m|\boldsymbol{o}_t, \boldsymbol{\lambda}) \log p(\boldsymbol{o}_t, m|\boldsymbol{\lambda}) , \tag{4.2}$$

where M denotes the number of mixtures .

Following equations are common for all the adaptation techniques and we will refer to them in the consequent text. Let ω_m, $\boldsymbol{\mu}_m$, $\boldsymbol{\sigma}_m^2$ and

$$\gamma_m(t) = p(m|\boldsymbol{o}_t, \boldsymbol{\lambda}) = \frac{\omega_m p(\boldsymbol{o}_t|m, \boldsymbol{\lambda})}{\sum_{m=1}^{M} \omega_m p(\boldsymbol{o}_t|m, \boldsymbol{\lambda})} \tag{4.3}$$

be the $m-th$ mixture weight, mean, variance and posterior probability, respectively. Let $c_m = \sum_{t=1}^{T} \gamma_m(t)$ be the soft count of mixture m and let the vector $\boldsymbol{\varepsilon}_m(\boldsymbol{o}) = \left[\sum_{t=1}^{T} \gamma_m(t)\boldsymbol{o}_t \right] \cdot c_m^{-1}$ be the average of features in frames which align to mixture m. Note: $\boldsymbol{\sigma}_m^2 = \mathrm{diag}(\boldsymbol{C}_m)$ is the diagonal of the covariance matrix $\boldsymbol{C}_m$.

4.1 Maximum Aposteriori Probability (MAP)

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. The new parameters $\bar{\boldsymbol{\lambda}} = \{\bar{\omega}_m, \bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{C}}_m\}_{m=1}^{M}$ are acquired according to adaptation formulas

$$\bar{\omega}_m = [\alpha_m c_m / T + (1 - \alpha_m)\omega_m]\,\chi\,, \tag{4.4}$$

$$\bar{\boldsymbol{\mu}}_m = \alpha_m \boldsymbol{\varepsilon}_m(\boldsymbol{o}) + (1 - \alpha_m)\boldsymbol{\mu}_m\,, \tag{4.5}$$

$$\bar{\boldsymbol{C}}_m = \alpha_m \boldsymbol{\varepsilon}_m(\boldsymbol{o}\boldsymbol{o}^{\mathrm{T}}) + (1 - \alpha_m)(\boldsymbol{\sigma}_m^2 + \boldsymbol{\mu}_m\boldsymbol{\mu}_m^{\mathrm{T}}) - \bar{\boldsymbol{\mu}}_m\bar{\boldsymbol{\mu}}_m^{\mathrm{T}}\,, \tag{4.6}$$

where

$$\alpha_m = \frac{c_m}{c_m + \tau}\,. \tag{4.7}$$

Hence, α_m represents the adaptation coefficient 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 mixture till the mixture parameters change (they shift in the direction of new parameters) [Ale05]. χ is a normalization factor, which guarantees that all the new weights of one mixture sum to unity.

4.2 Maximum Likelihood Linear Regression (MLLR)

In contrast to the method of MAP adaptation, where large amount of data is needed, MLLR reduces the number of available model parameters using clustering (commonly used is the regression tree) of similar components. Parameters from the same cluster $K_n, n = 1, \ldots, N$, share one transformation matrix. Thus, less data are needed for MLLR to be effective. If GMMs are assumed, the auxiliary function (4.2) that has to be maximized can be, after some mathematical treatments, written as

$$Q(\boldsymbol{\lambda}, \bar{\boldsymbol{\lambda}}) = const - \frac{1}{2}\sum_m \sum_t \gamma_m(t)\left[const_m + \log|\bar{\boldsymbol{C}}_m| + (\boldsymbol{o}_t - \bar{\boldsymbol{\mu}}_m)^{\mathrm{T}}\bar{\boldsymbol{C}}_m^{-1}(\boldsymbol{o}_t - \bar{\boldsymbol{\mu}}_m)\right]\,, \tag{4.8}$$

where $\bar{\boldsymbol{\lambda}} = \{\omega_m, \bar{\boldsymbol{\mu}}_m, \bar{\boldsymbol{C}}_m\}_{m=1}^{M}$, thus mixture weights are not adapted, and $const$, $const_m$ denote constants. The given task is to find linear transformations for GMM means and GMM variances that would maximize the auxiliary function specified in (4.8).

Mean is transformed according to the formula

$$\bar{\boldsymbol{\mu}}_m = \boldsymbol{A}_{(n)}\boldsymbol{\mu}_m + \boldsymbol{b}_{(n)} = \boldsymbol{W}_{(n)}\boldsymbol{\xi}_m\,, \tag{4.9}$$

$$\boldsymbol{W}_{(n)} = [\boldsymbol{A}_{(n)}, \boldsymbol{b}_{(n)}] \tag{4.10}$$

where $\boldsymbol{\mu}_m$ is the original mean of the m-th mixture, $\bar{\boldsymbol{\mu}}_m$ is the new, adapted mean, $\boldsymbol{A}_{(n)}$ is the regression matrix, $\boldsymbol{b}_{(n)}$ is the additive vector that corresponds to the $n - th$ cluster K_n

and $\boldsymbol{\xi}_m = [\boldsymbol{\mu}_m^{\mathrm{T}}, 1]^{\mathrm{T}}$ is the original mean extended by 1. The part of the auxiliary function (4.8) that changes with the current transform $\boldsymbol{W}_{(n)}$ can be written as [PS06]

$$Q_{\boldsymbol{W}_{(n)}} = const - \sum_{m \in K_n} c_m \sum_{i=1}^{I} \frac{(\boldsymbol{w}_{(n)i}^{\mathrm{T}} \boldsymbol{\xi}_m)^2 - 2(\boldsymbol{w}_{(n)i}^{\mathrm{T}} \boldsymbol{\xi}_m)\varepsilon_{mi}(\boldsymbol{o})}{\sigma_{mi}^2} , \tag{4.11}$$

where the column vector $\boldsymbol{w}_{(n)i}$ equals the transpose of the i-th row of $\boldsymbol{W}_{(n)}$ and I is the dimension of feature vectors. Equation (4.11) can be further rearranged to the form

$$Q_{\boldsymbol{W}_{(n)}} = \boldsymbol{w}_{(n)i}^{\mathrm{T}} \boldsymbol{k}_{(n)i} - 0.5\, \boldsymbol{w}_{(n)i}^{\mathrm{T}} \boldsymbol{G}_{(n)i} \boldsymbol{w}_{(n)i} , \tag{4.12}$$

where

$$\boldsymbol{k}_{(n)i} = \sum_{m \in K_n} \frac{c_m \boldsymbol{\xi}_m \varepsilon_{mi}(\boldsymbol{o})}{\sigma_{mi}^2} \tag{4.13}$$

and

$$\boldsymbol{G}_{(n)i} = \sum_{m \in K_n} \frac{c_m \boldsymbol{\xi}_m \boldsymbol{\xi}_m^{\mathrm{T}}}{\sigma_{mi}^2} . \tag{4.14}$$

And finally the maximization of equation (4.12) gives us the updating formulas

$$\frac{\partial Q(\boldsymbol{\lambda}, \bar{\boldsymbol{\lambda}})}{\partial \boldsymbol{W}_{(n)}} = 0 \Rightarrow \boldsymbol{w}_{(n)i} = \boldsymbol{G}_{(n)i}^{-1} \boldsymbol{k}_{(n)i} . \tag{4.15}$$

The transformation equations for covariance matrices can be derived in a similar way. They will be not discussed here as they are no important for further explanations and can be found in [Gal97].

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

Compared to MLLR the transformation is applied on the feature space (feature MLLR) instead of on model parameters. The auxiliary function (4.2) changes to [Gal97]

$$\begin{aligned} Q(\boldsymbol{\lambda}, \bar{\boldsymbol{\lambda}}) = const - \frac{1}{2} \sum_m \sum_t \gamma_m(t) [const_m + \log |\boldsymbol{C}_m| - \log\, |\boldsymbol{A}_{(n)}|^2 + \\ + (\bar{\boldsymbol{o}}_t - \boldsymbol{\mu}_m)^{\mathrm{T}} \boldsymbol{C}_m^{-1} (\bar{\boldsymbol{o}}_t - \boldsymbol{\mu}_m)] . \end{aligned} \tag{4.16}$$

The feature vectors are transformed according to the formula

$$\bar{\boldsymbol{o}}_t = \boldsymbol{A}_{(n)} \boldsymbol{o}_t + \boldsymbol{b}_{(n)} = \boldsymbol{A}_{(n)c}^{-1} \boldsymbol{o}_t + \boldsymbol{A}_{(n)\mathrm{c}}^{-1} \boldsymbol{b}_{(n)\mathrm{c}} = \boldsymbol{W}_{(n)} \boldsymbol{\xi}(t) , \tag{4.17}$$

where $\boldsymbol{W}_{(n)} = [\boldsymbol{A}_{(n)}, \boldsymbol{b}_{(n)}]$ stands for the transformation matrix corresponding to the $n - th$ cluster K_n, $\boldsymbol{\xi}(t) = [\boldsymbol{o}_t^{\mathrm{T}}, 1]^{\mathrm{T}}$ represents the extended feature vector. It can be shown [Gan05]

that the transformation performed on features may be replaced by an *equivalent transformation* performed on model means and covariances utilizing matrices $\boldsymbol{A}_{(n)\mathrm{c}}$ and $\boldsymbol{b}_{(n)\mathrm{c}}$. Hence, model parameters can be transformed using formulas

$$\bar{\boldsymbol{\mu}}_m = \boldsymbol{A}_{(n)\mathrm{c}}\boldsymbol{\mu}_m - b_{(n)\mathrm{c}} \,, \tag{4.18}$$

and

$$\bar{\boldsymbol{C}}_m = \boldsymbol{A}_{(n)\mathrm{c}}\boldsymbol{C}_m\boldsymbol{A}_{(n)\mathrm{c}}^{\mathrm{T}} \,. \tag{4.19}$$

This method is known as Constrained MLLR (CMLLR), because the same transformation matrix is used both for means and 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 auxiliary function (4.16) to the form [PS06]

$$Q_{\boldsymbol{W}_{(n)}}(\boldsymbol{\lambda}, \bar{\boldsymbol{\lambda}}) = \log|\boldsymbol{A}_{(n)}| - \sum_{i=1}^{I} \boldsymbol{w}_{(n)i}^{\mathrm{T}}\boldsymbol{k}_i - 0.5\, \boldsymbol{w}_{(n)i}^{\mathrm{T}}\boldsymbol{G}_{(n)i}\boldsymbol{w}_{(n)i} \,, \tag{4.20}$$

where

$$\boldsymbol{k}_{(n)i} = \sum_{m\in K_n} \frac{c_m\mu_{mi}\boldsymbol{\varepsilon}_m(\boldsymbol{\xi})}{\sigma_{mi}^2} \,, \tag{4.21}$$

$$\boldsymbol{G}_{(n)i} = \sum_{m\in K_n} \frac{c_m\boldsymbol{\varepsilon}_m(\boldsymbol{\xi}\boldsymbol{\xi}^{\mathrm{T}})}{\sigma_{mi}^2} \,, \tag{4.22}$$

$$\boldsymbol{\varepsilon}_m(\boldsymbol{\xi}) = \left[\boldsymbol{\varepsilon}_m^{\mathrm{T}}(\boldsymbol{o}), 1\right]^{\mathrm{T}} \,, \tag{4.23}$$

and

$$\boldsymbol{\varepsilon}(\boldsymbol{\xi}\boldsymbol{\xi}^{\mathrm{T}})_m = \begin{bmatrix} \boldsymbol{\varepsilon}_m(\boldsymbol{o}\boldsymbol{o}^{\mathrm{T}}) & \boldsymbol{\varepsilon}_m(\boldsymbol{o}) \\ \boldsymbol{\varepsilon}_m(\boldsymbol{o})^{\mathrm{T}} & 1 \end{bmatrix} \,. \tag{4.24}$$

In order to find the solution of equation (4.20) 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 $\alpha_{(n)} = \boldsymbol{w}_{(n)i}^{\mathrm{T}}\boldsymbol{v}_{(n)i}$. After the maximization of the auxiliary function (4.20) we receive

$$\boldsymbol{w}_{(n)i} = \boldsymbol{G}_{(n)i}^{-1}\left(\frac{\boldsymbol{v}_{(n)i}}{\alpha_{(n)}} + \boldsymbol{k}_{(n)i}\right) \,, \tag{4.25}$$

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

$$\beta_{(n)}\alpha_{(n)}^2 - \alpha_{(n)}\, \boldsymbol{v}_{(n)i}^{\mathrm{T}}\boldsymbol{G}_{(n)i}^{-1}\boldsymbol{k}_{(n)i} - \boldsymbol{v}_{(n)i}^{\mathrm{T}}\boldsymbol{G}_{(n)i}^{-1}\boldsymbol{v}_{(n)i} = 0 \,, \tag{4.26}$$

where

$$\beta_{(n)} = \sum_{m\in K_n}\sum_{t}\gamma_m(t) \,. \tag{4.27}$$

Because of equation (4.26) two different solutions $\boldsymbol{w}_{(n)i}^{1,2}$ are obtained in (4.25). The one that maximizes the auxiliary function (4.20) 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 mixture [Gan05]. Hence, for CMLLR we get the log likelihood

$$\log p\left(\boldsymbol{o}_t|\boldsymbol{\mu}_m, \boldsymbol{C}_m, \boldsymbol{A}_{(n)c}, \boldsymbol{b}_{(n)c}\right) \quad = \quad \log \mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{A}_{(n)c}\boldsymbol{\mu}_m - \boldsymbol{b}_{(n)c}, \boldsymbol{A}_{(n)c}\boldsymbol{C}_m\boldsymbol{A}_{(n)c}^{\mathrm{T}}\right), \quad (4.28)$$

but for fMLLR we get

$$\log p\left(\boldsymbol{o}_t|\boldsymbol{\mu}_m, \boldsymbol{C}_m, \boldsymbol{A}_{(n)}, \boldsymbol{b}_{(n)}\right) \quad = \quad \log \mathcal{N}\left(\boldsymbol{A}_{(n)}\boldsymbol{o}_t + \boldsymbol{b}_{(n)}; \boldsymbol{\mu}_m, \boldsymbol{C}_m\right) \; + \; 0.5 \log |\boldsymbol{A}_{(n)}|^2 \; . \quad (4.29)$$

The estimation of $\boldsymbol{W}_{(n)}$ is an iterative procedure, 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) [PS06].

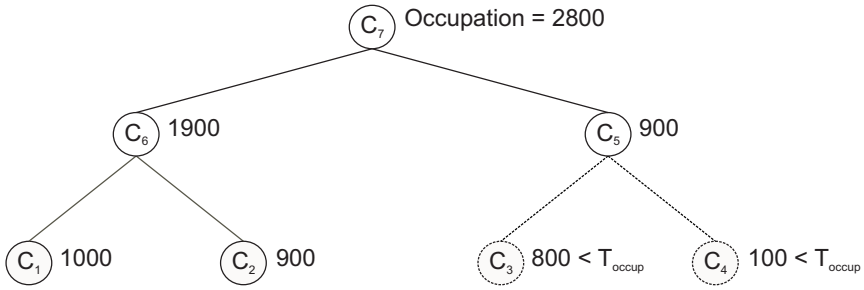

Figure 4.1: An example of a binary regression tree. Numerical values represent occupation counts of nodes (clusters). Nodes C_3 and C_4 have occupations lesser then the occupation threshold $T_{occup} = 850$, therefore all the mixture parameters belonging to C_3 and C_4 will share the transformation matrix defined for node C_5.

4.4 Regression Classes for MLLR

The benefit of MLLR like methods is the possibility to cluster similar mixture 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 [YEG+06], optimal clustering techniques trying to maximize the likelihood of the adaptation data [Gal96] 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 mixtures in each of the clusters – *cluster occuppations*, where an empirical threshold has to be set. For example, let us analyze the tree depicted in Figure 4.1 with four leaves $\{C_1, C_2, C_3, C_4\}$. The clusters C_3 and C_4 have small occupation counts (lower then the threshold $T_{occup} = 850$), 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.

4.5 Conclusion and Remarks

After inspection of equations (4.4) - (4.7), (4.13) - (4.15), (4.21) - (4.22), (4.25) - (4.27) one can notice that all the adaptation formulas depend on the data only through statistics $\gamma_m(t)$, c_m, $\boldsymbol{\varepsilon}_m(\boldsymbol{o})$ [PS06]. Thus, the adaptation process can be divided into two stages. In the first stage, the three mentioned statistics are accumulated till the data are introduced to the algorithm and in the second stage the transformation matrices for the desired type of adaptation are computed. There is no need to specify the adaptation type in advance and it can be defined after the adaptation data have been accumulated. The computational costs are also low as the updating matrices $\boldsymbol{G}_{(n)i}, \boldsymbol{k}_{(n)i}$ have not to be recalculated in each iteration (they are computed only once at the end of the adaptation).

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 5

Hybrid Generative/Discriminative Models

Recently, a huge progress was done in the use of SVM as an discriminative trainer in the speaker verification. 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. One of the important questions that can be stated is whether there are problems that can be solved only by discriminative methods, whereas any generative method would fail. It is shown in [LSS07] that such problems exist. However, the idea is to combine both approaches to gain a superior performance of such a hybrid system. The experiments are very promising and prove that hybrid models based on SVMs act at least as good as the generative ones and in many cases even better [JH99, MHV03].

5.1 Dynamic Kernels

In order to use a SVM, the data that have to be classified need to be of fixed dimension. Nevertheless, the speech utterances are usually parametrized as variable length sequences. First method solving this problem was developed by Jaakola and Hausler [JH99] and is known as Fisher kernel (see Subsection 5.1.2). Over the time many other kernels were proposed and successfully tested. Also a basic hierarchical structure of kernels was proposed, where the dynamic kernel (also known as sequence kernel) can be further divided into *parametric* and *derivative* kernels [LG07].

Dynamic kernels have the form

$$K(\boldsymbol{O}^i, \boldsymbol{O}^j, \boldsymbol{\theta}) = \psi(\boldsymbol{O}^i; \boldsymbol{\theta})^{\mathrm{T}} \psi(\boldsymbol{O}^j; \boldsymbol{\theta}), \tag{5.1}$$

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

$$K(\boldsymbol{O}^i, \boldsymbol{O}^j, \boldsymbol{\theta}) = \psi(\boldsymbol{O}^i; \boldsymbol{\theta})^{\mathrm{T}} \boldsymbol{G}\, \psi(\boldsymbol{O}^j; \boldsymbol{\theta}), \tag{5.2}$$

where

$$\boldsymbol{G}^{-1} = \varepsilon\{(\psi(\boldsymbol{O}; \boldsymbol{\theta}) - \boldsymbol{\mu}_\psi)^{\mathrm{T}} (\psi(\boldsymbol{O}; \boldsymbol{\theta}) - \boldsymbol{\mu}_\psi)\}, \tag{5.3}$$

$$\boldsymbol{\mu}_\psi = \varepsilon\{\psi(\boldsymbol{O}; \boldsymbol{\theta})\}, \tag{5.4}$$

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

5.1.1 Parametric Kernels

Parametric kernels deal with high-dimensional vectors called supervectors. The recent feature extraction methods have focused mainly on GMM means and adaptation matrices derived from speaker independent models according to the speaker dependent data (see Chapter 4). A very common extraction technique concatenates the GMM means so that a high-dimensional supervector is formed. To ensure correspondence between two adjacent mixtures of two different GMMs and same dimensionality of subsequently formed supervectors, it is appropriate to MAP-adapt speaker's Gaussian mixture models from an Universal Background Model (UBM). Thus, the mapping $\psi(\cdot)$ has the form

$$\psi(\boldsymbol{O}; \boldsymbol{\lambda}) = [\boldsymbol{\mu}_1^{\mathrm{T}}, \cdots, \boldsymbol{\mu}_i^{\mathrm{T}}, \cdots, \boldsymbol{\mu}_N^{\mathrm{T}}]^{\mathrm{T}}, \tag{5.5}$$

where $\boldsymbol{\mu}_i$ are the MAP-adapted GMM means (mixture weights and covariance matrices remain the same), N is the number of mixtures in the UBM and $\boldsymbol{\lambda}$ is the set of parameters of the generative model.

A different mapping proposed in [CSRS06a] takes into consideration the difference between means of the model of a new speaker and UBM means, and can be written as

$$\begin{aligned} \psi_{\text{diff}}(\boldsymbol{O}; \{\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}}\}) &= \psi(\boldsymbol{O}; \boldsymbol{\lambda}) - \psi(\,\cdot\,; \hat{\boldsymbol{\lambda}}) \\ &= [(\boldsymbol{\mu}_1 - \hat{\boldsymbol{\mu}}_1)^{\mathrm{T}}, \cdots, (\boldsymbol{\mu}_i - \hat{\boldsymbol{\mu}}_i)^{\mathrm{T}}, \cdots, (\boldsymbol{\mu}_N - \hat{\boldsymbol{\mu}}_N)^{\mathrm{T}}]^{\mathrm{T}}, \end{aligned} \tag{5.6}$$

where $\hat{\boldsymbol{\mu}}_i$ represents a mean of the original UBM and the notation $\psi(\,\cdot\,; \hat{\boldsymbol{\lambda}})$ represents the mapping of UBM means, hence a mapping same for any sequence $\boldsymbol{O}$ related to a specific speaker. Loosely speaking, the mapping $\psi(\,\cdot\,; \hat{\boldsymbol{\lambda}})$ is independent on the sequence of feature vectors $\boldsymbol{O}$ related to one speaker.

In [Cam02] the proposed mapping, denoted as Generalized Linear Discriminant Sequence (GLDS), is based on a vector function that transforms directly the feature vectors. The mapping has the form

$$\psi(\boldsymbol{O};\boldsymbol{\varphi}(\cdot)) = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\varphi}(\boldsymbol{o}_t)\,, \tag{5.7}$$

$$\boldsymbol{\varphi}(\boldsymbol{o}_t) = [\varphi_1(\boldsymbol{o}_t),\ldots,\varphi_j(\boldsymbol{o}_t),\ldots,\varphi_J(\boldsymbol{o}_t)]^{\mathrm{T}}\,, \tag{5.8}$$

where $\boldsymbol{\varphi}(\cdot)$ represents an expansion of the input space into a vector of scalar functions, $\varphi_j : R^m \mapsto R$, m is the dimension of the input space and J is the dimension of the vector function $\boldsymbol{\varphi}(\cdot)$. In [Cam02] the considered feature expansion consisted of monomials up to the $l-th$ order, e.g. for monomials up to the second order of the feature vector $\boldsymbol{o} = \{o_1, o_2, \ldots, o_I\}$ we get

$$\boldsymbol{\varphi}_{l=2}(\boldsymbol{o}) = [1, o_1, \ldots, o_I, o_1^2, o_1o_2, \ldots, o_1o_I, o_2^2, o_2o_3, \ldots, o_2o_I, o_I^2]\,, \tag{5.9}$$

where $\dim(\boldsymbol{o}) = I$ and $\dim(\boldsymbol{\varphi}_l(\boldsymbol{o})) = [(I+l)!]/[\,I!\,l!\,]$. After substituting (5.9) into (5.7) one can notice, that the mapping (5.7) comprises first and second-order moments – the means and correlations of feature vectors (higher moments will be obtained for higher order monomials) [LYKZ08]. The main disadvantage of such a mapping is that the function $\boldsymbol{\varphi}_l(\boldsymbol{o})$ has to be applied on each of the feature vectors, what may be quite expensive especially for longer utterances. Further generalizations of GLDS and associated kernels allowing not only polynomial degrees but also infinite dimensional expansions were studied in [LDB06, LDB07].

In [LYLK07] the vector function $\boldsymbol{\varphi}(\boldsymbol{o})$ was chosen as

$$\boldsymbol{\varphi}_{\mathrm{PS}}(\boldsymbol{o}_t) = [\gamma_1(\boldsymbol{o}_t),\ldots,\gamma_i(\boldsymbol{o}_t),\ldots,\gamma_N(\boldsymbol{o}_t)]\,, \tag{5.10}$$

where $\gamma_m(\boldsymbol{o}_t)$ is the $m-th$ mixture's posterior specified in (4.3). The mapping based on $\boldsymbol{\varphi}_{\mathrm{PS}}(\boldsymbol{o_t})$ is called the Probabilistic Sequence (PS) mapping and has the same form as (5.7) except the vector function, hence

$$\psi(\boldsymbol{O};\boldsymbol{\varphi}_{\mathrm{PS}}(\cdot)) = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\varphi}_{\mathrm{PS}}(\boldsymbol{o}_t)\,. \tag{5.11}$$

When using such a mapping, it is useful to cover the part of the acoustic space of interest with well distributed Gaussian mixtures over this area - e.g. to utilize some anchor models chosen from the background population of speakers [LYKZ08].

Another kind of supervector extraction is based on MLLR transformation matrices computed according to Section 4.2 and given in equation (4.10). The rows of such transformation matrices are subsequently concatenated into one high-dimensional supervector. The mapping $\psi(\cdot)$ can be expressed as

$$\psi(\boldsymbol{O};\{\boldsymbol{W}_{(k)}\}_{k=1}^{K}) = [\boldsymbol{w}_{(1)1}^{\mathrm{T}},\cdots,\boldsymbol{w}_{(1)I}^{\mathrm{T}},\boldsymbol{w}_{(2)1}^{\mathrm{T}},\ldots,\boldsymbol{w}_{(2)I}^{\mathrm{T}},\ldots,\boldsymbol{w}_{(K)1}^{\mathrm{T}},\ldots,\boldsymbol{w}_{(K)I}^{\mathrm{T}}]^{\mathrm{T}}\,, \tag{5.12}$$

where I 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 $\boldsymbol{W}_{(k)}$.

5.1.1.1 Mean supervector based kernels

A basic, frequently used and well-working kernel is the one in (5.2), 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 some cases even replaced by the identity matrix [WR02].

In [CSR06] two kernels were proposed – Supervector Linear Kernel (SLK) and L^2 Inner Product Kernel (L^2IPK). The SLK is based on the approximated Kullback-Leibler Divergence (KLD) and in [MHV03] even the unapproximated KLD was utilized – the kernels were denoted as Probabilistic Distance Kernels (PDKs).

Supervector Linear Kernel (SLK) was derived from KLD. To ensure the validity of Mercer's condition (see (3.38)) an approximation of KLD [Do03] was computed. Assuming diagonal covariances the resulting kernel can be written in the form

$$K_{\mathrm{SLK}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \sum_{i=1}^{N} \omega_i (\boldsymbol{\mu}_i^{spk1})^{\mathrm{T}} \boldsymbol{C}_i^{-1} \boldsymbol{\mu}_i^{spk2} = \psi(\boldsymbol{O}^{spk1}; \boldsymbol{\lambda})^{\mathrm{T}}\, \boldsymbol{\Omega}\tilde{\boldsymbol{G}}\, \psi(\boldsymbol{O}^{spk2}; \boldsymbol{\lambda})\,, \tag{5.13}$$

$$\tilde{\boldsymbol{G}} : \mathrm{diag}(\tilde{\boldsymbol{G}}) = [\mathrm{diag}(\boldsymbol{C}_1^{-1}), \ldots, \mathrm{diag}(\boldsymbol{C}_i^{-1}), \ldots, \mathrm{diag}(\boldsymbol{C}_N^{-1})]^{\mathrm{T}}\,, \tag{5.14}$$

$$\boldsymbol{\Omega} : \mathrm{diag}(\boldsymbol{\Omega}) = [\underbrace{\omega_1, \ldots, \omega_1}_{I}, \ldots, \underbrace{\omega_i, \ldots, \omega_i}_{I}, \ldots, \underbrace{\omega_N, \ldots, \omega_N}_{I}]^{\mathrm{T}}\,, \tag{5.15}$$

where $\omega_i, \boldsymbol{\mu}_i, \boldsymbol{C}_i$ are related to the GMM (see (3.14)) and $\tilde{\boldsymbol{G}}$, $\boldsymbol{\Omega}$ are zero matrices with diagonals specified in (5.14), (5.15), respectively. The matrix $\boldsymbol{C}_i$ 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 (5.3) and $\boldsymbol{\Omega}$ represents an additional weighting factor, where $I = \dim(\boldsymbol{o}) = \dim(\boldsymbol{\mu})$. There are several advantages of such a kernel [Cam02]. The matrix $\boldsymbol{R} = \boldsymbol{\Omega}\tilde{\boldsymbol{G}}$ can be factored using Cholesky decomposition yielding $\boldsymbol{R} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}$ and all the supervectors can be transformed before the SVM training as $\boldsymbol{Y} = \boldsymbol{U}\psi(\boldsymbol{O}; \boldsymbol{\lambda})$. Hence the kernel takes the form

$$K_{\mathrm{SLK}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}\,, \tag{5.16}$$

thus there is no need to multiply supervectors with $\boldsymbol{R}$ whenever the kernel function is evaluated, therefore the computational costs during the training will be significantly reduced. Furthermore, the model compaction technique can be applied, thus the SVM decision hyperplane (3.35) with L support vectors can be computed in advance and the final decision function can be expressed as

$$f(\boldsymbol{O_j}) = \left(\boldsymbol{\Omega}\tilde{\boldsymbol{G}} \sum_{k=1}^{L} \alpha_k y_k{}^{\mathrm{T}} \psi(\boldsymbol{O_k}; \boldsymbol{\lambda}) \right) \psi(\boldsymbol{O_j}; \boldsymbol{\lambda}) + b\,. \tag{5.17}$$

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

$$K_{\mathrm{L^2IPK}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \int_{R^n} g_{spk1}(\boldsymbol{o}) g_{spk2}(\boldsymbol{o})\, \mathrm{d}\boldsymbol{o}. \tag{5.18}$$

The closed form solution has the form

$$K_{\mathrm{L^2IPK}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \sum_{i=1}^{N}\sum_{j=1}^{N} \omega_i \omega_j \mathcal{N}\left(\boldsymbol{\mu}_i^{spk1} - \boldsymbol{\mu}_j^{spk2}; \mathbf{0}, \boldsymbol{C}_i + \boldsymbol{C}_j\right), \tag{5.19}$$

where $\mathbf{0}$ 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, (5.19) can be simplified to the form

$$K_{\mathrm{L^2IPK}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \sum_{i=1}^{N}\sum_{j=1}^{N} \omega_i^2 \mathcal{N}(\boldsymbol{\mu}_i^{spk1} - \boldsymbol{\mu}_j^{spk2}; \mathbf{0}, 2\boldsymbol{C}_i). \tag{5.20}$$

Probabilistic Distance Kernels (PDKs) were proposed in [MHV03] and as already mentioned, they utilize even the unapproximated KLD. The symmetricity of such a dissimilarity measure was preserved using

$$D(p(\boldsymbol{o}|\boldsymbol{\lambda}_1) \,||\, p(\boldsymbol{o}|\boldsymbol{\lambda}_2)) = \int_{-\infty}^{\infty} p(\boldsymbol{o}|\boldsymbol{\lambda}_1) \log\left(\frac{p(\boldsymbol{o}|\boldsymbol{\lambda}_1)}{p(\boldsymbol{o}|\boldsymbol{\lambda}_2)}\right) \mathrm{d}\boldsymbol{o} + \int_{-\infty}^{\infty} p(\boldsymbol{o}|\boldsymbol{\lambda}_2) \log\left(\frac{p(\boldsymbol{o}|\boldsymbol{\lambda}_2)}{p(\boldsymbol{o}|\boldsymbol{\lambda}_1)}\right) \mathrm{d}\boldsymbol{o}. \tag{5.21}$$

and the validity of the kernel was ensured with the help of the equation

$$K_{\mathrm{PDK}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \mathrm{e}^{-const_1 D(p(\boldsymbol{o}|\boldsymbol{\lambda}_1) \,||\, p(\boldsymbol{o}|\boldsymbol{\lambda}_2)) + const_2}, \tag{5.22}$$

where $const_1, const_2$ represent scale and shift factors, respectively, involved because of stability reasons. In the case of GMM there is no analytic solution of (5.21) and some numerical approximations have to be employed (e.g. [Do03]), but in the case of a full covariance GMM with a single mixture, the distance in (5.21) can be computed directly yielding

$$\begin{aligned} D(\mathcal{N}(\cdot;\boldsymbol{\mu}_1,\boldsymbol{C}_1) \,||\, \mathcal{N}(\cdot;\boldsymbol{\mu}_2,\boldsymbol{C}_2)) = \mathrm{tr}\left(\boldsymbol{C}_1\boldsymbol{C}_2^{-1}\right) + \mathrm{tr}\left(\boldsymbol{C}_1^{-1}\boldsymbol{C}_2\right) - 2I + \\ + \mathrm{tr}\left[\left(\boldsymbol{C}_1^{-1} + \boldsymbol{C}_2^{-1}\right)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^{\mathrm{T}}\right], \end{aligned} \tag{5.23}$$

where $I = \dim(\boldsymbol{o})$. Thus, not only mean vectors are involved, but also covariance matrices (however just for single mixture models). It is easy to see, that for $\boldsymbol{C} = \boldsymbol{C}_1 = \boldsymbol{C}_2$ and $\boldsymbol{C}$ assuming diagonal, the distance (5.23) degenerates to

$$D(\mathcal{N}(\cdot;\boldsymbol{\mu}_1,\boldsymbol{C}) \,||\, \mathcal{N}(\cdot;\boldsymbol{\mu}_2,\boldsymbol{C})) = d(\boldsymbol{\mu}_1,\boldsymbol{\mu}_2;\boldsymbol{C}) = 2\left\|\boldsymbol{C}^{-\frac{1}{2}}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^{\mathrm{T}}\right\|^2. \tag{5.24}$$

It can be proven (see [Do03]), that for two multi-mixture GMMs $g_1(\boldsymbol{o})$ and $g_2(\boldsymbol{o})$, the KLD $D(g_1(\boldsymbol{o})||g_2(\boldsymbol{o})) \leq \sum_{i=1}^{N} \omega_i d(\boldsymbol{\mu}_i^1, \boldsymbol{\mu}_i^2; \boldsymbol{C}_i)$. Hence, the KLD is upper bounded by the sum of distances of two adjacent mixtures. Based on the last observation another kernel was proposed in [DC06]. It can be expressed as

$$\begin{aligned} K_{\mathrm{PDK2}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) &= \mathrm{e}^{-0.5\sum_{i=1}^{N} \omega_i d(\boldsymbol{\mu}_i^{spk1}, \boldsymbol{\mu}_i^{spk2}; \boldsymbol{C}_i)} \\ &= \mathrm{e}^{-\sum_{i=1}^{N} \omega_i \left\|\boldsymbol{C}_i^{-\frac{1}{2}}(\boldsymbol{\mu}_i^{spk1} - \boldsymbol{\mu}_i^{spk2})^{\mathrm{T}}\right\|^2} \\ &= \mathrm{e}^{-\left\|\left(\boldsymbol{\Omega}\tilde{\boldsymbol{G}}\right)^{\frac{1}{2}}(\psi(\boldsymbol{O}^{spk1};\boldsymbol{\lambda}) - \psi(\boldsymbol{O}^{spk2};\boldsymbol{\lambda}))^{\mathrm{T}}\right\|^2}, \end{aligned} \tag{5.25}$$

where N is the number of UBM mixtures, $\omega_i, \boldsymbol{C}_i$ are the $i-th$ UBM mixture's weight and covariance matrix, respectively, and $\boldsymbol{\mu}_i^{spk1}, \boldsymbol{\mu}_i^{spk2}$ are adjacent, speaker dependent, MAP-adapted means. Matrices $\boldsymbol{\Omega}$ and $\tilde{\boldsymbol{G}}$ were specified in (5.15) and (5.14), respectively.

5.1.1.2 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.

The kernel function considered can be a simple linear inner product (5.1) of MLLR based supervectors as used in [SFK+05], nevertheless one could also transform the means according to the formula (4.9) and utilize kernels described in Subsection 5.1.1.1.

More sophisticated and more interesting kernel approach was proposed in [KC07]. It arises from (5.13), where means $\boldsymbol{\mu}_i$ are transformed according to equation (4.9), hence

$$K(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \sum_{i=1}^{N} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}}(\boldsymbol{A}\boldsymbol{\mu}_i + \boldsymbol{b})\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}}(\boldsymbol{H}\boldsymbol{\mu}_i + \boldsymbol{d})\right), \tag{5.26}$$

where $[\boldsymbol{A}, \boldsymbol{b}]$ and $[\boldsymbol{H}, \boldsymbol{d}]$ are global transformation matrices for speakers $spk1$ and $spk2$, respectively, and $\boldsymbol{\Delta}_i = \omega_i \boldsymbol{C}_i^{-1}$. After expanding equation (5.26) we get

$$\begin{aligned} K(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = &\sum_{i=1}^{N} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{A}\boldsymbol{\mu}_i\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{H}\boldsymbol{\mu}_i\right) \quad + \sum_{i=1}^{N} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{A}\boldsymbol{\mu}_i\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{d}\right) + \\ &+ \sum_{i=1}^{N} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{b}\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{H}\boldsymbol{\mu}_i\right) \quad + \sum_{i=1}^{N} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{b}\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{d}\right). \end{aligned} \tag{5.27}$$

Let us have a look on the first term in (5.27). Some notations shall be stated, $\mathrm{tr}(\boldsymbol{A})$ 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 $\boldsymbol{\Delta}_i$, I 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 $\boldsymbol{C}_i$)

$$\begin{aligned} \sum_{i=1}^{N} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{A}\boldsymbol{\mu}_i\right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{H}\boldsymbol{\mu}_i\right) &= \sum_{i=1}^{N} \mathrm{tr}\left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{A}\boldsymbol{\mu}_i\boldsymbol{\mu}_i^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{\Delta}_i^{\frac{1}{2}}\right) = \sum_{i=1}^{N} \mathrm{tr}\left(\boldsymbol{\Delta}_i\boldsymbol{A}\boldsymbol{\mu}_i\boldsymbol{\mu}_i^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\right) = \\ &= \sum_{i=1}^{N} \mathrm{tr}\left[\left(\sum_{k=1}^{I} \Delta_{ik}\boldsymbol{e}_k\boldsymbol{e}_k^{\mathrm{T}}\right)\boldsymbol{A}\boldsymbol{\mu}_i\boldsymbol{\mu}_i^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\right] = \\ &= \sum_{i=1}^{N}\sum_{k=1}^{I} \mathrm{tr}\left[\boldsymbol{e}_k\boldsymbol{e}_k^{\mathrm{T}}\boldsymbol{A}\left(\Delta_{ik}\boldsymbol{\mu}_i\boldsymbol{\mu}_i^{\mathrm{T}}\right)\boldsymbol{H}^{\mathrm{T}}\right] = \\ &= \sum_{k=1}^{I} \mathrm{tr}\left[\boldsymbol{e}_k^{\mathrm{T}}\boldsymbol{A}\left(\sum_{i=1}^{N} \Delta_{ik}\boldsymbol{\mu}_i\boldsymbol{\mu}_i^{\mathrm{T}}\right)\boldsymbol{H}^{\mathrm{T}}\boldsymbol{e}_k\right] = \\ &= \sum_{k=1}^{I} \boldsymbol{a}_k^{\mathrm{T}}\left(\sum_{i=1}^{N} \Delta_{ik}\boldsymbol{\mu}_i\boldsymbol{\mu}_i^{\mathrm{T}}\right)\boldsymbol{h}_k = \\ &= \sum_{k=1}^{I} \boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{R}_k\boldsymbol{h}_k\,. \end{aligned} \tag{5.28}$$

All the other terms in (5.27) can be rearranged in a similar fashion, what results in

$$\sum_{i=1}^{N}\left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{A}\boldsymbol{\mu}_i\right)^{\mathrm{T}}\left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{d}\right)=\sum_{k=1}^{I}d_k\boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{r}_k\,, \tag{5.29}$$

$$\sum_{i=1}^{N}\left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{b}\right)^{\mathrm{T}}\left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{H}\boldsymbol{\mu}_i\right)=\sum_{k=1}^{I}b_k\boldsymbol{r}_k^{\mathrm{T}}\boldsymbol{h}_k\,, \tag{5.30}$$

$$\sum_{i=1}^{N}\left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{b}\right)^{\mathrm{T}}\left(\boldsymbol{\Delta}_i^{\frac{1}{2}}\boldsymbol{d}\right)=\sum_{k=1}^{I}b_kd_k\delta_k\,, \tag{5.31}$$

where $\boldsymbol{r}_k=\sum_{i=1}^{N}\Delta_{ik}\boldsymbol{\mu}_i$ and $\delta_k=\sum_{i=1}^{N}\Delta_{ik}$. Now, the kernel (5.26) can be rewritten as

$$\begin{aligned}K(\boldsymbol{O}^{spk1},\boldsymbol{O}^{spk2})&=\sum_{k=1}^{I}\boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{R}_k\boldsymbol{h}_k+d_k\boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{r}_k+b_k\boldsymbol{r}_k^{\mathrm{T}}\boldsymbol{h}_k+b_kd_k\delta_k=\\&=\psi(\boldsymbol{O}^{spk1};[\boldsymbol{A},\boldsymbol{b}])^{\mathrm{T}}\boldsymbol{Q}\,\psi(\boldsymbol{O}^{spk2};[\boldsymbol{H},\boldsymbol{d}])\,.\end{aligned} \tag{5.32}$$

The matrix $\boldsymbol{Q}$ is symmetric, positive-definite (the kernel satisfies the Mercer's condition (3.38)) as it arises from (5.13) and consists of I blocks of size $(I+1)\times(I+1)$, where each block $\boldsymbol{Q}_k$ can be expressed as

$$\boldsymbol{Q}_k=\begin{pmatrix}\boldsymbol{R}_k & \boldsymbol{r}_k\\ \boldsymbol{r}_k^{\mathrm{T}} & \delta_k\end{pmatrix}\,. \tag{5.33}$$

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 square root of $\boldsymbol{Q}$ and thus apply the model compaction technique as discussed in Section 5.1.1.1.

It should be also noted, that the dimension of MLLR based supervectors ($I\times(I+1)$; $I=\dim(\boldsymbol{o})$) is in comparison with dimension of mean supervetors ($I\times N$) often significantly lesser, as the number N of Gaussian mixtures in the UBM is often high.

5.1.1.3 Multi-class MLLR kernels

The extension to the multi-class case is straightforward. The regression tree is involved, thus several transformation matrices are computed at a time. It should be noted, that at the supervector construction, some matrices may occur repeatedly in cases when two separate classes with insufficient amount of data descend from the same parent class (see Figure 4.1). The crucial problem is the construction of the regression tree. The approaches in Section 4.4 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 [SFK+05], where the regression tree is designed according to broad phonetic classes depicted in Figure 5.1. Each of the classes contains a set of mixtures for which a separate transformation

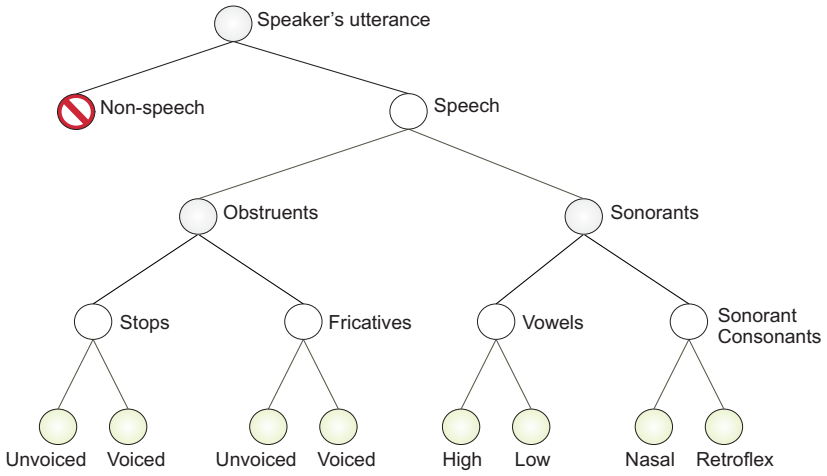

Figure 5.1: Regression tree based on phonetic classes.

matrix will be computed. The use of an text-independent GMM is now unfeasible – one cannot decide which mixture 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 4.2 and the regression tree depicted in Figure 5.1. Hence, the resulting matrices will correspond to proposed phonetic broad classes.

Another method was presented in [KC08]. It uses an N-mixture UBM with 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 5.1). The proposed UBM has the form

$$g(\boldsymbol{x}) = \vartheta_S \sum_{i=1}^{N/2} \omega_i \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{C}_i) + \vartheta_O \sum_{i=N/2+1}^{N} \omega_i \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{C}_i) \,, \tag{5.34}$$

where ϑ_S, ϑ_O are the weights for mixture components associated to sonorants and obstruents, respectively. They are calculated as the percentage of frames assigned to particular clusters. Each of the $N/2$ Gaussian mixtures 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 (5.34). Note that UBM weights have to be rescaled to sum to unity. All three models (one N-mixture UBM and two $N/2$-mixture models) are employed in the adaptation phase. Firstly an occupation threshold T_{occup} has to be specified as in Section 4.4. 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 then T_{occup}, the $N/2$-mixture models (trained for appropriate clusters) are adapted. For classes not satisfying the criterion, the N-mixture UBM 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 the LVCSR system adaptation, but it is much more easier to control the influence of mixtures in each of the phonetic classes. Generally, the number of mixtures associated to each class can be unequal (different from $N/2$), 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 5.1.1.2. One can consider only a simple linear inner product of MLLR based supervectors, or transform the model according to the formula (4.9) and use kernels from Subsection 5.1.1.1, or extend the approach described at the end of Subsection 5.1.1.2 as was done in [KC08]. The extension is straightforward if UBMs in the form of equation (5.34) are used. Nevertheless, with some effort the method could be easily extended also to cases when the LVCSR system is in use.

Let's have a look on UBMs defined as in (5.34). The kernel function (5.26) takes now the form

$$\begin{aligned} K(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) &= \vartheta_{\mathrm{S}} K_{\mathrm{S}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) + \vartheta_{\mathrm{O}} K_{\mathrm{O}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \\ &= \vartheta_{\mathrm{S}} \sum_{i=1}^{N/2} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} (\boldsymbol{A}_{\mathrm{S}} \boldsymbol{\mu}_i + \boldsymbol{b}_{\mathrm{S}}) \right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} (\boldsymbol{H}_{\mathrm{S}} \boldsymbol{\mu}_i + \boldsymbol{d}_{\mathrm{S}}) \right) + \\ &+ \vartheta_{\mathrm{O}} \sum_{i=N/2+1}^{N} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} (\boldsymbol{A}_{\mathrm{O}} \boldsymbol{\mu}_i + \boldsymbol{b}_{\mathrm{O}}) \right)^{\mathrm{T}} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} (\boldsymbol{H}_{\mathrm{O}} \boldsymbol{\mu}_i + \boldsymbol{d}_{\mathrm{O}}) \right). \end{aligned} \tag{5.35}$$

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

$$K_{\mathrm{S}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \psi_{\mathrm{S}}(\boldsymbol{O}^{spk1}; [\boldsymbol{A}_{\mathrm{S}}, \boldsymbol{b}_{\mathrm{S}}])^{\mathrm{T}} \boldsymbol{Q}_{\mathrm{S}}\, \psi_{\mathrm{S}}(\boldsymbol{O}^{spk2}; [\boldsymbol{H}_{\mathrm{S}}, \boldsymbol{d}_{\mathrm{S}}]), \tag{5.36}$$

$$K_{\mathrm{O}}(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \psi_{\mathrm{O}}(\boldsymbol{O}^{spk1}; [\boldsymbol{A}_{\mathrm{O}}, \boldsymbol{b}_{\mathrm{O}}])^{\mathrm{T}} \boldsymbol{Q}_{\mathrm{O}}\, \psi_{\mathrm{O}}(\boldsymbol{O}^{spk2}; [\boldsymbol{H}_{\mathrm{O}}, \boldsymbol{d}_{\mathrm{O}}]), \tag{5.37}$$

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 as in (5.33). The final kernel results in

$$K(\boldsymbol{O}^{spk1}, \boldsymbol{O}^{spk2}) = \begin{bmatrix} \psi_{\mathrm{S}}(\boldsymbol{O}^{spk1})^{\mathrm{T}} & \psi_{\mathrm{O}}(\boldsymbol{O}^{spk1})^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \vartheta_{\mathrm{S}} \boldsymbol{Q}_{\mathrm{S}} & \boldsymbol{0} \\ \boldsymbol{0} & \vartheta_{\mathrm{O}} \boldsymbol{Q}_{\mathrm{O}} \end{bmatrix} \begin{bmatrix} \psi_{\mathrm{S}}(\boldsymbol{O}^{spk2}) \\ \psi_{\mathrm{O}}(\boldsymbol{O}^{spk2}) \end{bmatrix}, \tag{5.38}$$

where the conditional part of $\psi_{\mathrm{S}}(\cdot)$, $\psi_{\mathrm{O}}(\cdot)$ was omitted because of lucidity. Properties of $\boldsymbol{Q}$ mentioned at the end of Section 5.1.1.2 are preserved. It should be stated, that the two approaches, either the use of the kernel (5.38) or the use of the kernel (5.13), where the means are transformed according to the formula (4.9), 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 [KC08] (this is useful especially when LVCSR systems are utilized).

5.1.2 Derivative Kernels

The derivative kernels are based on the work of Jaakkola and Haussler [JH99], who made the first connection between generative and discriminative models at all. Further investigations were carried out by Smith and Gales. They have proposed generalizations in the form

of score spaces defined by the mapping

$$\psi_{\tilde{F}}^{f}(\boldsymbol{O};\{\boldsymbol{\lambda}_q\}_{q=1}^{Q}) = \psi_{\tilde{F}} f(\{p(\boldsymbol{O}|\boldsymbol{\lambda}_q)\}_{q=1}^{Q}) \,, \tag{5.39}$$

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, it determines the form of the output of the set of generative models and is mapped by the score operator $\tilde{F}$ to a fixed-dimensional score space $\psi_{\tilde{F}}^{f}(\boldsymbol{O};\{\boldsymbol{\lambda}_q\}_{q=1}^{Q})$ [SG02]. 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 [Smi03]. The score space (5.39) for $Q = 1$, known as Log-Likelihood Score (LLS) space, can be expressed in the form

$$\psi_{\nabla}^{\mathrm{LLS}}(\boldsymbol{O};\hat{\boldsymbol{\lambda}},\rho) = \frac{1}{T}\left[\nabla_{\boldsymbol{\lambda}}^{(0,\rho)} \ln p(\boldsymbol{O}|\boldsymbol{\lambda})\Big|_{\hat{\boldsymbol{\lambda}}}\right] = \frac{1}{T}\begin{bmatrix} \ln p(\boldsymbol{O}|\hat{\boldsymbol{\lambda}}) \\ \nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{O}|\boldsymbol{\lambda})\Big|_{\hat{\boldsymbol{\lambda}}} \\ \vdots \\ \mathrm{vec}\left(\nabla_{\boldsymbol{\lambda}}^{\rho} \ln p(\boldsymbol{O}|\boldsymbol{\lambda})\Big|_{\hat{\boldsymbol{\lambda}}}\right) \end{bmatrix} , \tag{5.40}$$

where the term $\frac{1}{T}$ was introduced because of the sequence length normalization with T equal the number of feature vectors in $\boldsymbol{O}$, 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

$$\psi_{\nabla}^{\mathrm{LLRS}}(\boldsymbol{O};\{\hat{\boldsymbol{\lambda}}_q\}_{q=1}^{2},\rho) = \frac{1}{T}\left[\nabla_{\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2}^{(0,\rho)} \ln \frac{p(\boldsymbol{O}|\boldsymbol{\lambda}_1)}{p(\boldsymbol{O}|\boldsymbol{\lambda}_2)}\Big|_{\substack{\boldsymbol{\lambda}_1=\hat{\boldsymbol{\lambda}}_1 \\ \boldsymbol{\lambda}_2=\hat{\boldsymbol{\lambda}}_2}}\right] = \frac{1}{T}\begin{bmatrix} \ln p(\boldsymbol{O}|\hat{\boldsymbol{\lambda}}_1) - \ln p(\boldsymbol{O}|\hat{\boldsymbol{\lambda}}_2) \\ \nabla_{\boldsymbol{\lambda}_1} \ln p(\boldsymbol{O}|\boldsymbol{\lambda}_1)\big|_{\hat{\boldsymbol{\lambda}}_1} \\ -\nabla_{\boldsymbol{\lambda}_2} \ln p(\boldsymbol{O}|\boldsymbol{\lambda}_2)\big|_{\hat{\boldsymbol{\lambda}}_2} \\ \vdots \\ \mathrm{vec}\left(\nabla_{\boldsymbol{\lambda}_1}^{\rho} \ln p(\boldsymbol{O}|\boldsymbol{\lambda}_1)\Big|_{\hat{\boldsymbol{\lambda}}_1}\right) \\ -\mathrm{vec}\left(\nabla_{\boldsymbol{\lambda}_2}^{\rho} \ln p(\boldsymbol{O}|\boldsymbol{\lambda}_2)\Big|_{\hat{\boldsymbol{\lambda}}_2}\right) \end{bmatrix} . \tag{5.41}$$

The mapping used by Jaakkola and Haussler is a special case of (5.40) when only the first derivative is considered and is also known as Fisher mapping or Fisher score. Its expected value with respect to the observation $\boldsymbol{o}$ is zero and the mapping can be written in the form

$$\psi_{\nabla}^{\mathrm{Fisher}}(\boldsymbol{O};\hat{\boldsymbol{\lambda}}) = \frac{1}{T}\nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{O}|\boldsymbol{\lambda})\Big|_{\hat{\boldsymbol{\lambda}}} . \tag{5.42}$$

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

$$\nabla_{\boldsymbol{\mu}_i} \ln p(\boldsymbol{O}|\boldsymbol{\lambda}) = \sum_{t=1}^{T} \gamma_i(t) \boldsymbol{C}_i^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_i)^{\mathrm{T}} = \eta_i (\varepsilon_i(\boldsymbol{o}) - \boldsymbol{\mu}_i)^{\mathrm{T}} \boldsymbol{C}_i^{-1} \,, \tag{5.43}$$

$$\begin{aligned} \nabla_{\boldsymbol{C}_i} \ln p(\boldsymbol{O}|\boldsymbol{\lambda}) &= \sum_{t=1}^{T} \frac{\gamma_i(t)}{2} \left[-\boldsymbol{C}_i^{-1} + \boldsymbol{C}_i^{-1} (\boldsymbol{\mu}_i - \boldsymbol{o}_t)(\boldsymbol{\mu}_i - \boldsymbol{o}_t)^{\mathrm{T}} \boldsymbol{C}_i^{-1} \right] \\ &= \frac{\eta_i}{2} \left[-\boldsymbol{C}_i^{-1} + \boldsymbol{C}_i^{-1} [\hat{\boldsymbol{C}}_i + (\boldsymbol{\mu} - \varepsilon_i(\boldsymbol{o}))(\boldsymbol{\mu} - \varepsilon_i(\boldsymbol{o}))^{\mathrm{T}}] \boldsymbol{C}_i^{-1} \right] \,, \end{aligned} \tag{5.44}$$

where $\gamma_i(t)$ is the $i-th$ mixture's posterior specified in (4.3), $\eta_i = \sum_{t=1}^{T} \gamma_i(t)$, the vector $\boldsymbol{\varepsilon}_i(\boldsymbol{o}) = \left[\sum_{t=1}^{T} \gamma_i(t) \boldsymbol{o}_t \right] \cdot \eta_i^{-1}$ and the matrix $\hat{\boldsymbol{C}}_i = \boldsymbol{\varepsilon}_i(\boldsymbol{o}\boldsymbol{o}^{\mathrm{T}}) - \boldsymbol{\varepsilon}_i^2(\boldsymbol{o})$. The operator ε_i can be also seen as a statistical expectation of features aligning to the $i-th$ mixture of a GMM. It is very easy to see that the derivatives vanish when the data perfectly fits the model (e.g. for ML-estimates of $\boldsymbol{\lambda}$; $\hat{\boldsymbol{\lambda}} = \arg\max_{\boldsymbol{\lambda}} \{\ln p(\boldsymbol{O}; \boldsymbol{\lambda})\}$), thus $\boldsymbol{\mu}_i \equiv \boldsymbol{\varepsilon}_i(\boldsymbol{o})$ and $\hat{\boldsymbol{C}}_i \equiv \boldsymbol{C}_i$ for $i = 1, \ldots, N$, where N is the number of GMM mixtures. When data lie away from the model, absolute values of gradients in (5.43) and (5.44) increase, on the other hand, the closer are the data to the model, the smaller are the gradient absolute values (under the assumption, that the amount of data does not vary significantly – consider the influence of η_i – however, this problem is solved involving the normalization term 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 comes very handy. LLRS space takes into consideration not only deviation from the speaker's GMM, but also from some other model $\hat{\boldsymbol{\lambda}}_2$ – often represented by the UBM. UBM is 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 (5.44) 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 vec$(\cdot)$. If only diagonal covariances are assumed, the function vec$(\cdot)$ may be replaced by the function diag$(\cdot)$. Note that the term η_i associated with each mixture can be regarded as an alternative to the normalization term T.

Basic Derivative Kernel (BDK) is related to the kernel in equation (5.2). When utilizing the Fisher mapping (5.42), the normalization matrix $\boldsymbol{G}^{-1}$ from (5.3) 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 $\boldsymbol{\lambda}$ [UC04]. 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 hand over to the training are build upon MAP-adapted model of the current (target) speaker and all the utterances of all the participating speakers (for LLRS space in (5.41) also UBM is involved). Loosely speaking, no other models than the target speaker model are involved in the training phase of one SVM model.

Generalized Derivative Kernel (GDK) was introduced in [LG08a]. It is based on the GLDS mapping discussed in Section 5.1.1 and on the Fisher score, where only derivatives with respect to the mean are considered. Hence, (5.42) can be rewritten to the form

$$\psi_{\nabla}^{\mathrm{GDK}}(\boldsymbol{O};\tilde{\boldsymbol{\lambda}}) = \frac{1}{\tilde{\eta}_i}\sum_{t=1}^{T}\tilde{\gamma}_i(t)\tilde{\boldsymbol{C}}_i^{-1}(\boldsymbol{\varphi}(\boldsymbol{o}_t)-\tilde{\boldsymbol{\mu}}_i)^{\mathrm{T}}\,, \tag{5.45}$$

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 (5.8). The normalization term changes now to $\tilde{\eta}_m = \sum_{t=1}^{T}\tilde{\gamma}_m(t)$. After substituting (5.45) into (5.2) we get

$$\begin{aligned} K(\boldsymbol{O}^i,\boldsymbol{O}^j,\tilde{\boldsymbol{\lambda}}) &= \psi_{\nabla}^{\mathrm{GDK}}(\boldsymbol{O}^i;\tilde{\boldsymbol{\lambda}})^{\mathrm{T}}\,\boldsymbol{G}\,\psi_{\nabla}^{\mathrm{GDK}}(\boldsymbol{O}^j;\tilde{\boldsymbol{\lambda}}) \\ &= \sum_{m=1}^{M}\frac{1}{\tilde{\eta}_{im}\tilde{\eta}_{jm}}\sum_{t=1}^{T_i}\sum_{s=1}^{T_j}\tilde{\gamma}_m(t)\tilde{\gamma}_m(s)k_m(\boldsymbol{o}_{it},\boldsymbol{o}_{js}) \end{aligned} \tag{5.46}$$

$$k_m(\boldsymbol{o}_i,\boldsymbol{o}_j) = (\boldsymbol{\varphi}(\boldsymbol{o}_i)-\tilde{\boldsymbol{\mu}}_m)^{\mathrm{T}}\tilde{\boldsymbol{C}}_m^{-1}\boldsymbol{G}_m\tilde{\boldsymbol{C}}_m^{-1}(\boldsymbol{\varphi}(\boldsymbol{o}_j)-\tilde{\boldsymbol{\mu}}_m)\,. \tag{5.47}$$

There are several issues concerning the solution of (5.46). 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 (5.9)). Therefore, several approximations and adjustments are needed, for details see [LG08a].

5.2 On Training of Dynamic Kernels

In previous sections the construction of supervectors and subsequently, proposal of kernels dealing with such high-dimensional features were discussed. Now, we will take a closer look at the treatment of input data and at some normalization techniques related to supervectors. The result of the SVM classifier is in fact not a probability, it is the distance from the decision boundary (the exact geometrical distance will be obtained as in (3.23)). The probability outputs are very handy in situations when several systems are combined in order to obtain an overall decision. Therefore, some methods concerning probabilistic outputs for SVM will be given at the end of this section.

At the beginning, it should be noted that the hybrid modelling employing SVM and high-dimensional vectors demands a huge amount of background data involving as much speakers as possible. The background data are partitioned into two groups (recall that the SVM is a binary classifier). One is used to train the UBM and the content of the other group (optionally bigger, because of high-dimensional mappings) is used as negative examples (impostors) in the SVM training – *one-against-all* training. Nevertheless, in [FNG01] another way was followed. To train the SVM the *All-Pairs* technique described in [HT98] was applied. Thus, for each pair of speakers a binary classifier was trained and the pairwise results were combined at the end. Hence, during the training of a decision boundary between two different classes the data from other classes were ignored. Obviously, the main disadvantage of such an approach is the need to evaluate the output of each classifier, what may be intractable in cases when many speakers are engaged (a solution intended for further work is given in Section 6).

5.2.1 Dealing with Unbalanced Data

The problem concerns the fact that the amount of positive and negative examples (speaker vs. impostors), submitted to the SVM training, is often highly disproportional (assuming no pairwise coupling). In principle, each of the mappings described in sections 5.1.1 and 5.1.2 generates for each speaker's utterance only one supervector. In the case, when the utterance is long enough, it can be split into several parts and each of the parts can be mapped to the new feature space separately. However, this does not solve the problem completely as the number of impostor speakers is supposed to be huge, because of the one-against-all training. The weighting has to be implemented directly in the formulation of the SVM problem stated in (3.27) [MBJ99], thus

$$\text{minimize} \quad \left[\frac{1}{2} \|\boldsymbol{w}\|^2 + C^+ \sum_{i:y_i=+1} \xi_i + C^- \sum_{j:y_j=-1} \xi_j \right] \tag{5.48}$$

subject to

$$\forall i : y_i(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{o}_i + b) \geq 1 - \xi_i \,, \tag{5.49}$$

where $\boldsymbol{w}$ represents, in agreement with Section 3.3, the normal vector of the hyperplane $\mathcal{H}$ defined in (3.22). 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.

5.2.2 Normalization Techniques

At first, three techniques focused on feature vector normalization will be mentioned. These are especially helpful when high-dimensional vectors and their dot products are utilized in the SVM training. One of the well known problems in speaker verification systems is the choice of the verification threshold. The task is influenced by many disturbing factors making more difficult to correctly tune the threshold value. Namely: not precisely trained speaker models, similar voices of speakers, inter and intra speaker variabilities, environment conditions and many others, which reflect themselves more or less in the verification score. Hence, subsequently, two common methods of score normalization will be reviewed. And at the end a method dealing with channel variability compensation will be discussed.

Feature Space Whitening (FSW) deals with the fact that the support vector machine is not invariant to linear transformations. Consider a set of N two dimensional vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. Let $\mu_{1,2}$ and $\sigma^2_{1,2}$ 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 $\sigma_1^2 >> \sigma_2^2$ leads to domination of the first dimension in the dot product of the two vectors from $\boldsymbol{X}$ reducing the dimensionality of the space to one [WR05]. Hence, it is desirable to normalize elements of the vectors to zero mean and unit variance – whiten the data. Note that the unit variance normalization is performed employing the matrix $\boldsymbol{G}$ from (5.3) and the kernel function in the form of equation (5.2).

Rank Normalization (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 [SFK+05]. The procedure is as follows:

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

It is useful to further normalize the vectors along each dimension to a suitable interval, e.g. $[0, 1]$. The disadvantage of RN are its computational costs.

Spherical Normalization (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 [WR05]. 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 (3.36) badly conditioned or, in the worst case, even singular – especially for polynomial kernels with high powers [WC00]. 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 or score space vectors. 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 [WR05] has the form

$$\phi(\boldsymbol{x}) = \frac{1}{\sqrt{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} + d^2}} \begin{bmatrix} \boldsymbol{x} \\ d \end{bmatrix}, \tag{5.50}$$

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 $\|\boldsymbol{x}\|_2$ insted of SN, but this would lead to information loss (consider two distinct vectors $\boldsymbol{x}$ and $2\boldsymbol{x}$).

Zero and Test Normalizations are related to the verification score. The Zero Normalization (Z-norm) method was proposed by [Rey97] and its purpose is to ensure zero mean and unit standard deviation of scores for speech of impostor speakers. The Z-normalized score for the $s-th$ speaker and $i-th$ sequence of feature vectors $\boldsymbol{O}^i$ (parametrized utterance) will be computed according to

$$L_{\mathrm{Z}}(\boldsymbol{O}^i|s) = \frac{L(\boldsymbol{O}^i|s) - \mu_{\mathrm{I}}}{\sigma_{\mathrm{I}}}, \tag{5.51}$$

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 as follows. The speaker model is tested against impostor utterances yielding a set of scores related to the given speaker model. This set is used to compute the Z-norm parameters. Note that the parameters for Z-norm can be estimated offline during speaker model training.

The Test Normalization (T-norm) introduced in [Auc00] is computed using the same formula as Z-norm. The difference consists in estimation of μ_{I} and σ_{I}. A set of impostor models is chosen beforehand. The T-norm parameters are then estimated from the set of scores obtained during the testing phase, where all the impostor models are tested 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 [RDL+92], 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. [SR05].

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 [CVC+06].

Nuisance Attribute Projection (NAP) was presented in [SQC04] and is suited for SVMs. It reduces the influence of the channel variability in the speaker recognition task by projecting out the supervector dimensions which are mostly vulnerable to changes of environment conditions. The projection can be written in the form

$$\bar{\boldsymbol{x}} = \left(\boldsymbol{I} - \boldsymbol{\Gamma}\boldsymbol{\Gamma}^{\mathrm{T}}\right)\boldsymbol{x}\,, \tag{5.52}$$

where in our case $\boldsymbol{x} = \psi(\boldsymbol{O};\boldsymbol{\theta})$ defined in (5.1) with $\dim(\boldsymbol{x}) = I_x$, $\boldsymbol{I}$ is the $I_x \times I_x$ identity matrix, $\boldsymbol{\Gamma}$ denotes $I_x \times I_p$ matrix, where I_p is equal to the number of dimensions that we wish to project out. $\boldsymbol{\Gamma}$'s columns are formed by eigenvectors $\boldsymbol{v}_i$ corresponding to the I_p largest eigenvalues λ_i obtained when solving the eigenvalue problem

$$\boldsymbol{A}(\mathrm{DIAG}(\boldsymbol{W}\boldsymbol{1}) - \boldsymbol{W})\boldsymbol{A}^{\mathrm{T}}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i\,, \tag{5.53}$$

where $\boldsymbol{A} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_x}]$ represents the data matrix with N_x vectors ordered in columns, $\boldsymbol{1}$ is a column vector of all ones, the function $\mathrm{DIAG}(\boldsymbol{x})$ creates a zero matrix with diagonal equal to entries of the vector $\boldsymbol{x}$ and $\boldsymbol{W} = [W_{ij}]$ is a $N_x \times N_x$ symmetric, weight matrix. It can be chosen in several ways, e.g.

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

In the first case minimization of cross-channel distances is demanded, in the latter case minimization of the session variability is desired [SCB05]. The problem with $\boldsymbol{W}_{\mathrm{channel}}$ is that it tries to put together also different speakers, although recorded on different channels. This can be solved by centralizing the supervectors of one speaker. Denote $\boldsymbol{A}_s$ as the part of $\boldsymbol{A}$ containing N_s supervectors pertaining to the speaker s. The centralization can be done with the use of $N_s \times N_s$ matrix $\boldsymbol{J}_s$ as $\boldsymbol{A}_{cs} = \boldsymbol{A}_s\boldsymbol{J}_s$, where $\boldsymbol{A}_{cs}$ is the centralized matrix of the speaker s and

$$\boldsymbol{J}_s = \boldsymbol{I} - \frac{1}{N_s}\boldsymbol{1}\,\boldsymbol{1}^{\mathrm{T}}\,. \tag{5.54}$$

Hence, the speaker identity should be normalized out, but several recordings of sessions for one speaker are needed to correctly estimate the mean value of speaker supervectors. Nevertheless, the centralization is quite useful also for $\boldsymbol{W}_{\mathrm{session}}$ [CSRS06b].

Some remarks should be stated. The eigenvalue problem (5.53) is symmetric. Hence, the eigenvectors have unit lengths and they form even a orthonormal basis. If the problem would not be symmetric, the columns of $\mathbf{\Gamma}$ would need to be normalized to unit lengths (for details see the definition of the problem in [SQC04]). Note, that the dimension of the feature vector after being projected (utilizing (5.52)) is preserved, only the rank (also called *corank*) of the matrix $\boldsymbol{A}$ decreased – the vector $\boldsymbol{x}$ is projected onto a plane in the high-dimensional space. The formulation of NAP was originally proposed for SVMs, but recently some attempts have been made to extended it also to conventional GMMs [VM08]. And finally, it should be stated that NAP demands a huge labeled set of background data to be trained.

5.2.3 Probabilistic Outputs for SVM

In this subsection several methods for adjustment of output of SVM classifiers will be discussed. The output of the support vector machine can be thought of as a distance from the decision boundary, which is in principle unbounded. Thus, it can take any values in the interval $(-\infty, \infty)$. Nevertheless, if only a close set of trials is considered, we can easily identify the boundary values of the set of distances $D = \{d_i\}_{i=1}^{T_d}$. One trial represents one verification attempt, when the identity of a claimed speaker is tested against the identity of one reference speaker and d_i is one distance associated to one trial. The set D will be obtained after submitting all the trials to the verification system. The first presented method maps the set of results D onto the interval $[0, 1]$ utilizing a simple linear function in the form

$$\begin{aligned} D^{[0,1]} &= \{d_i : d_i^{[0,1]} = kd_i + b,\ \ i = 1, \ldots, T_d\}\,, \\ k &= (\max_d\{D\} - \min_d\{D\})^{-1}\,,\ \ b = -k \cdot \min_d\{D\}\,. \end{aligned} \tag{5.55}$$

An analogical approach to the rank normalization can be exploited too. Hence, the results are sorted and their values are replaced by their rank in the sorted list (subsequently normalized to the interval [0,1], i.e. percentile). However, when additional trials accrue, the results have to be remaped again as the boundary values of the set of distances may have changed.

Another, more sophisticated method, uses a GMM to rescale the results [KSKW07]. It estimates the posterior probabilities $p(f(x_i)|y_i = +1)$ and $p(f(x_i)|y_i = -1)$ utilizing ML approach and labeled data. The final decision is computed according to the Bayes rule

$$p(y_i = +1|f(x_i)) = \frac{p(f(x_i)|y_i = +1)P(y = +1)}{\sum_{j\in\{1,-1\}} p(f(x_i)|y_i = j)P(y = j)}\,, \tag{5.56}$$

where the function $f(\cdot)$ denotes the SVM decision boundary, y_i represents the label of class x_i and the priors $P(y = \pm 1)$ are estimated according to the training set. In [Pla99] the probability $p(y_i = +1|f(x_i))$ is estimated directly and has the form of a sigmoid function

$$p(y_i = +1|f(x_i)) = \frac{1}{1 + \exp(Af(x) + B)}\,, \tag{5.57}$$

where A, B control the slope and the position of the inflection point of the sigmoid, respectively. Both can be estimated via ML approach (see [Pla99]).

5.3 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. Recently, attempts to study the complementarity of parametric and derivative kernels have been carried out [LG07]. 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 5.1.2. 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. Note that when the generarative classifier in the form of GMM would be optimal (see Section 3.1), the information supported by the derivative mappings would be of no use.

Until now, the SVM was assumed as a tool utilizing proposed kernels. Nevertheless, through the time several other machines were developed and exploited. E.g. the Sparse Kernel Logistic Regression (SKLR) proposed in [KKS+06], which models directly the posterior probabilities of the class membership, or the learning algorithm in [SMW08], which has the advantage that it does not suffer from the strict Mercer's kernel restrictions defined in (3.38).

In this chapter only MLLR based kernels have been discussed, but generally, the 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 4.3 [FLBG07].

The last remark concerns the dimensionality of supervectors, which is high indeed (several thousands). Hence, methods for dimensionality reduction like Principial Component Analysis (PCA), Linear Discriminant Analysis (LDA) or Heteroscedastic LDA (HLDA) may come handy [QB02].

Chapter 6

Conclusion and Future Work

This report has focused on the process of augmentation of statistical models, especially on augmentation of GMMs. Two group of approaches were discussed. The first, denoted as *parametric kernels*, involved directly the parameters of statistical models or parameters related to the statistical model (transformation matrices acquired in the adaptation process). Supervectors were formed and mainly means were considered. However, the GLDS mapping (assuming order of monomials greater then two), or probabilistic distance kernels described at the beginning of Section 5.1.1, have utilized in a simple way also covariances. The latter group, *derivative kernels*, was based upon derivatives of the statistical model. The derivative mappings may be interpreted as a measure of errors related to data and model structure. As stated in Section 2, supervectors can be seen as higher level features situated in a high-dimensional feature space. Recalling Section 5.2.1, only a few supervectors (often only one) related to one speaker are extracted – mapped into the high dimensional space. Hence, tools as GMM or HMM that estimate class models are inappropriate. To cope with such a problem a high amount of background data (data from non-target speakers) is demanded to delimitate the speaker specific region. Subsequently, the SVM is used to define the boundaries between speaker data and background data that belong to all the other, non-target speakers.

The problem of hybrid modelling is a new one and has a great potential to be further developed. The basics have been already described in the last few years, but there are still issues to deal with. Let's mention some of them, most suitable for further work.

- As already pointed out, we are able to describe the shifting of each UBM mixture according to the speaker data. Another source of information, when thinking of augmentation of GMMs, would be the description of inner topology of one GMM that represents the class model of one particular speaker. The inner topology can be thought of as a description of the relation between pairs of mixtures (e.g. relation in the sense of direction or distance). To guarantee the consistency in the process of mixtures estimation in cases when distinct utterances originating from one speaker are modeled, an adaptation of the UBM has to be involved. The adaptation ensures same initial conditions in the training process and defines connections between mixtures in the adapted GMM and

the original UBM. Thus, it is easy to determine which two mixtures should be compared when considering two GMMs (both adapted from the same UBM).

- The theoretical background for derivative kernels has been introduced by Smith [Smi03]. There are many opportunities mainly in the choice of the score-operator $\tilde{F}$ (see (5.39)) not investigated yet. Similar theoretical background should be developed also for parametric kernels. By now, the GMM means are extracted in a raw form (they are only concatenated). However, the supervectors may be processed further to extract as many information as possible. E.g. the GLDS mapping (see beginning of Section 5.1.1) could be employed.

- In Section 5.2 two SVM training approaches were mentioned, the one-against-all training and All-Pairs technique. The main disadvantage of the All-Pairs technique are the computational costs (see Section 5.2). However, when one would suitably divide the background data into few separate clusters and a decision boundary would be computed only between the speaker model and data in each of the clusters, the number of classifiers (decision boundaries) would decrease. It can be anticipated that such boundaries would describe the speaker discrimination more properly. Such an approach could be further investigated in the future work.

- Other problem of main importance refers to the background data submitted to the training of the SVM classifier. The background data play a crucial role in the estimation of SVM parameters (see the discussion in the beginning of the conclusion). Hence, two problems has to be examined – what data should be used (*criterion*) and how should they be chosen (*method*)?

- After the discussion in the conclusion of Section 5 a tool/kernel exploiting the complementarity of derivative and parametric kernels would be of interest. Generally, a common tool for fusion of mappings on the kernel level is demanded. Some investigations were already carried out by Longworth and Gales [LG08b].

- Another improvement, suitable for the future work, concerns the GMM covariances and solves the following question. How could be the change in mixture covariances employed into the high dimensional mapping in an useful and comfort way?

- In Section 5.1.1.2 a kernel induced by MLLR transformation matrices was proposed. To fully exploit its possibilities, the approach should be extended also to the LVCSR systems.

- One should employ and investigate also other classifiers than SVM.

The above specified points are sorted in accordance to the priority for my future work. The approaches presented in this work were not discussed in much depth. The intention of the submitted report was to become familiar with the task of hybrid modelling and related methods. More detailed description of the whole problem and further investigations will be the subject of the PhD thesis.

Bibliography

[Ale05] A. Alexander. *Forensic automatic speaker recognition using bayesian interpretation and statistical compensation for mismatched conditions*. PhD thesis, Indian Institute of Technology, Madras, 2005.

[Auc00] R. Auckenthaler. Score normalization for text-independent speaker verification systems. *Digital Signal Processing*, 10(1-3):42–54, January 2000.

[BBF+04] 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*, 4:430–451, 2004.

[Bur98] J.C.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[Cam02] W.M. Campbell. Generalized linear discriminant sequence kernels for speaker recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP'02*, 1:I–161–I–164, 2002.

[CBW01] R. Collobert, S. Bengio, and C. Williamson. Svmtorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[CSR06] W.M. Campbell, D.E. Sturim, and D.A. Reynolds. Support vector machines using gmm supervectors for speaker verification support vector machines using gmm supervectors for speaker verification. *Signal Processing Letters, IEEE*, 13(5):308–311, 2006.

[CSRS06a] 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, 2006. ICASSP 2006 Proceedings*, 1:I–I, 2006.

[CSRS06b] 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. *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'06*, 1:I–I, 2006.

[CVC+06] D. Colibro, C. Vair, F. Castaldo, E. Dalmasso, and P. Laface. Speaker recognition using channel factors feature compensation. *EUSIPCO-2006*, 2006.

[DC06] N. Dehak and G. Chollet. Support vector gmms for speaker verification. *The Speaker and Language Recognition Workshop. IEEE Odyssey 2006*, pages 1–4, 2006.

[DHS00] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

[DLR77] 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)*, 39(1):1–38, 1977.

[Do03] M.N. Do. Fast approximation of kullback-leibler distance for dependence trees and hidden markov models. *Signal Processing Letters, IEEE*, 10(4):115–118, 2003.

[Čer01] Jan Černocký. Traps in all senses, report of post-doc research internship. Technical report, 2001.

[FLBG07] 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*, 4:53–56, 2007.

[FNG01] 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*, 1:417–420, 2001.

[Gal96] M.J.F. Gales. The generation and use of regression class trees for mllr adaptation. Technical report, Cambridge University Engineering Department, 1996.

[Gal97] M.J.F. Gales. Maximum likelihood linear transformation for hmm-based speech recognition. Technical report, Cambridge University Engineering Department, 1997.

[Gan05] J. Ganitkevitch. Speaker adaptation using maximum likelihood linear regression. Technical report, Rheinisch-Westfalische Technische Hochschule Aachen, 2005.

[Her90] H. Hermansky. Perceptual linear predictive (plp) analysis of speech. *The Journal of the Acoustical Society of America*, 87:1738–1752, 1990.

[HT98] T. Hastie and R. Tibshirani. Classification by pairwise coupling. In *The Annals of Statistics*, pages 507–513. MIT Press, 1998.

[JH99] T.S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *In Advances in Neural Information Processing Systems 11*, pages 487–493. MIT Press, 1999.

[Joa02] T. Joachims. SVM light, http://svmlight.joachims.org, 2002.

[KC07] Z.N. Karam and W.M. Campbell. A new kernel for svm mllr based speaker recognition. *Proc. Interspeech 2007, Antwerp, Belgium*, pages 290–293, 2007.

[KC08] 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*, pages 4117–4120, 2008.

[KKS+06] 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*, 153/2006:176–184, 2006.

[KSKW07] M. Katz, M. Schafföner, S.E. Krüger, and A. Wendemuth. Score calibrating for speaker recognition based on support vector machines and gaussian mixture models. In *SIP*, pages 139–144, 2007.

[LDB06] 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*, pages 1–5, 2006.

[LDB07] 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*, 15:2465–2475, 2007.

[LG07] C. Longworth and M.J.F. Gales. Parametric and derivative kernels for speaker verification. *Interspeech 2007*, pages 310–313, 2007.

[LG08a] C. Longworth and M.J.F. Gales. A generalised derivative kernel for speaker verification. *Proceedings of Interspeech, Brisbane*, pages 1385–1388, 2008.

[LG08b] C. Longworth and M.J.F Gales. Multiple kernel learning for speaker verification. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 1581–1584, 2008.

[LSS07] P.M. Long, A.R. Servedio, and H.U. Simon. Discriminative learning can succeed where generative learning fails. *Inf. Process. Lett.*, 103(4):131–135, August 2007.

[LYKZ08] K. Lee, C. You, T. Kinnunen, and D. Zhu. Characterizing speech utterances for speaker verification with sequence kernel svm. *Proceedings of Interspeech, Brisbane*, pages 1397–1400, 2008.

[LYLK07] K.A. Lee, C. You, H. Li, and T. Kinnunen. A gmm-based probabilistic sequence kernel for speaker verification. *Proc. Interspeech 2007*, pages 294–297, 2007.

[MBJ99] 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)*, pages 268 – 277, 1999.

[MHV03] 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.

[Pla99] J.C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74, 1999.

[PMMR07] J. Psutka, L. Müller, J. Matoušek, and V. Radová. *Mluvíme s počítačem česky*. ACADEMIA Praha, 2007.

[PS06] D. Povey and G. Saon. Feature and model space speaker adaptation with full covariance gaussians. In *Interspeech*, pages paper 2050–Tue2BuP.14, Pittsburgh, PA, USA, 2006.

[QB02] L. Quan and S. Bengio. Hybrid generative-discriminative models for speech and speaker recognition. Technical report, Dalle Molle Institute for Perceptual Artificial Intelligence, Martigny, Valais, Switzerland, 2002.

[Rab89] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RDL+92] A. Rosenberg, J. Delong, C. Lee, B. Juang, and F. Soong. The use of cohort normalized scores for speaker recognition. *ICSLP 1992*, pages 599–602, 1992.

[Rey92] D.A. Reynolds. *A Gaussian Mixture Modelling Approach to Text-independent Speaker Identification*. PhD thesis, Georgia Institute of Technology, 1992.

[Rey97] D.A. Reynolds. Comparison of background normalization methods for text-independent speaker verification. *EUROSPEECH-1997, 5th European Conference on Speech Communication and Technology*, 2:963–966, 1997.

[SCB05] 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*, 1:629–632, 2005.

[SFK+05] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman. Mllr transforms as features in speaker recognition. *Proc. Eurospeech, Lisbon*, pages 2425–2428, 2005.

[SG02] N.D. Smith and M.J.F Gales. Using svms to classify variable length speech patterns. Technical report, Cambridge University Engineering Department, 2002.

[Smi03] N.D. Smith. *Using Augmented Statistical Models and Score Spaces for Classification*. PhD thesis, University of Cambridge, 2003.

[SMW08] A. Stuhlsatz, H.G. Meier, and A. Wendemuth. Maximum margin classification on convex euclidean metric spaces. In *Computer Recognition Systems 2*, pages 216–223. 2008.

[SQC04] A. Solomonoff, C. Quillen, and W. Campbell. Channel compensation for svm speaker recognition. *Odyssey'04*, pages 219–226, 2004.

[SR05] 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*, 1:741–744, 2005.

[SS02] B. Schölkopf and A.J. Smola. *Learning with Kernels.* MIT Press, 2002.

[TK03] S. Theodoridis and K. Koutroumbas. *Pattern Recognition, Second Edition.* Elsevier Academic Press, 2003.

[UC04] G. Upton and I. Cook. *A Dictionary of Statistics.* Oxford University Press, 2004.

[VM08] B. Vesnicer and F. Mihelic. The likelihood ratio decision criterion for nuisance attribute projection in gmm speaker verification. *EURASIP Journal on Applied Signal Processing*, 2008, 2008.

[V.V95] V.Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.

[WC00] 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*, 2:775 – 784, 2000.

[WR02] V. Wan and S. Renals. Evaluation of kernel methods for speaker verification andidentification. *IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP'02*, 1:I–669– I–672, 2002.

[WR05] V. Wan and S. Renals. Speaker verification using sequence discriminant support vector machines. *IEEE Transactions on Speech and Audio Processing*, 13:203 – 210, 2005.

[YEG+06] 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, 2001-2006.

[ZGR+94] 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*, pages 55–58, 1994.