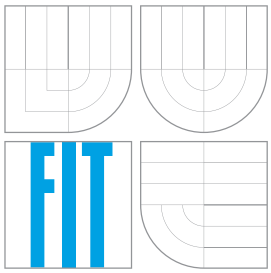
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ ŘEČNÍKA POMOCÍ TEMPORÁLNÍCH PŘÍZNAKŮ

SPEAKER RECOGNITION BASED ON LONG TEMPORAL CONTEXT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADEK FÉR

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Dr. Ing. JAN ČERNOCKÝ

BRNO 2014

Abstrakt

Tato práce se zabývá extrakcí vhodných příznaků pro rozpoznávání řečníka z delších časových úseků. Po představení současných technik pro extrakci takových příznaků navrhujeme a popisujeme novou metodu pracující v časovém rozsahu fonémů a využívající známou techniku i-vektorů. Velké úsilí bylo vynaloženo na nalezení vhodné reprezentace temporálních příznaků, díky kterým by mohly být systémy pro rozpoznávání řečníka robustnější, zejména modelování prosodie. Náš přístup nemodeluje explicitně žádné specifické temporální parametry řeči, namísto toho používá kookurenci řečových rámců jako zdroj temporálních příznaků. Tuto techniku testujeme a analyzujeme na řečové databázi NIST SRE 2008. Z výsledků bohužel vyplývá, že pro rozpoznávání řečníka tato technika nepřináší očekávané zlepšení. Tento fakt diskutujeme a analyzujeme ke konci práce.

Abstract

This work deals with temporal features for automated speaker recognition. We give overview of currently known temporal feature extraction methods and afterwards, we propose and preliminarily evaluate a general phoneme-level temporal feature extraction scheme based on factor analysis i-vector paradigm. Much effort has been made to reasonably represent temporal context and make speaker recognition systems more robust, namely speech prosody modeling. Our approach does not explicitly model any temporal parameters of speech, rather it uses the occurrence of neighboring frames as a source of temporal information. We test and analyze this method on standard evaluation database NIST SRE 2008. The results indicate, however, that for speaker recognition, no useful gain can be obtained using this technique. We describe and discuss this discovery at the end.

Klíčová slova

rozpoznávání řečníka, temporální příznaky, i-vektory, parametrizace řeči

Keywords

speaker recognition, temporal features, i-vectors, speech parametrization

Citace

Radek Fér: Speaker Recognition Based on Long Temporal Context, diplomová práce, Brno, FIT VUT v Brně, 2014

Rozpoznávání řečníka pomocí temporálních příznaků

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Dr. Drisse Matroufa a pana doc. Jana Černockého. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Radek Fér
July 30, 2014

Poděkování

Děkuji mému vedoucímu práce v Brně, panu doc. Janu Černockému za vedení práce a za zprostředkování mé stáže v Avignonu. Dále děkuji mému vedoucímu práce v Avignonu, panu Dr. Drissovi Matroufovi, za vedení práce a cenné rady a náměty, bez kterých by tato práce nevznikla.

Contents

Chapter 1

Introduction

1.1 Motivation

In the last decade, the speaker recognition technology has undertaken a big development. Performance of state-of-the-art systems has been drastically improved and the processing speed-up is by several orders of magnitude. More or less successful techniques were developed to make the technology robust against the biggest obstacle in the field of speaker recognition – session mismatch. The methods for fusing different systems were explored and heterogeneous systems are now de-facto the standard. Also, new speech processing toolkits and speech data were released, which makes the technology more available.

Many applications of speaker recognition technology can be found. Telephone companies, banks and others (including police and intelligence agencies of course) would like to use speaker recognition systems for different purposes: voice verified authentication, call tracking or automatic labeling of voice data (speaker diarization), etc. The development of speaker recognition technology goes side by side with the development of language identification (LID), so this can be also a motivation.

In this work, we develop a general data-driven temporal feature extraction technique. The scope of currently used features used in speech processing tasks is far from being truly general in a sense of explicitly emphasizing all the relevant aspects of speech signal. Instead, based on the knowledge, that some speech phenomena contain speaker discriminative information, different features are extracted from signal on such levels and information is merged by fusing the output scores. The scope of our approach is limited to the temporal information contained in speech segments of length around 100 ms.

1.1.1 Organization of this work

For this work to be self-introductory and comprehensible also for the people not familiar with speaker recognition, we present fundamental concepts of speaker recognition later in this chapter. The currently used and documented temporal features are described in Chapter 2. Afterwards, in Chapter 3, we give a short introduction to speaker modeling methods and channel compensation. In our work, we use heavily the concept of i-vectors, so this chapter is mainly written as an introduction to this topic. In Chapter 4, we present our proposal of feature extraction technique based on i-vector paradigm. Experimental results are presented in Chapter 5, with a separate discussion and work conclusions in Chapter 6.

1.2 Fundamental concepts of speaker recognition

Speaker recognition is very suitable for commercial biometric purposes because it does not need any additional hardware and in the case of text-independent recognition, it can be performed transparently, i.e. without user even knowing that verification of his voice is in progress. In this work, we deal with automated speaker verification (ASV). This is a problem of deciding the question „Is this a voice of person X?“. It differs slightly from the task of speaker identification, where the question is put in the following way: „For a given list of persons, who is speaking?“. The important difference between the two is that the speaker detection task does not depend on the number of target speakers.

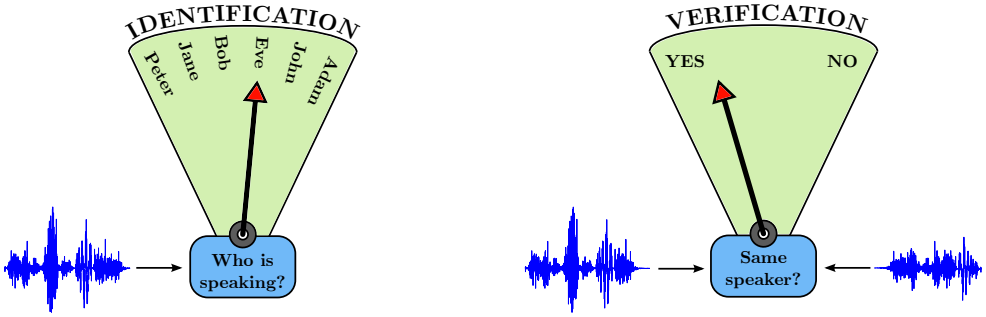


Figure 1.1: Automatic speaker identification (ASI) and verification (ASV).

1.2.1 Feature extraction

The first step for any speech-related pattern matching task is to extract *features* from the input signal in a form suitable to computers and classification algorithms. Humans, during their lives, learn differences in voice of other people using well-known characteristic features of individual, that can be obtained from his/her speech – pitch, timbre, speaking rate, intonation, selection of specific words, voice abnormalities etc. Ideal speaker recognition system should utilize all of them. The process of extraction of such information from raw speech is called feature extraction. This is not an easy task, because this information is mixed up with the other information present in speech signal, like speaker mood, channel information, linguistic content, noise and so on.

1.2.2 Speaker modeling

After that, different pattern matching algorithms are used to create speaker models, classify test utterances and, at the end, make decisions. The modeling techniques can be divided into generative and discriminative ones. Generative techniques model the distribution of training features, whereas the discriminative techniques do not make any assumption on the feature distribution and model rather the boundary between speakers. The examples of generative and discriminative models are Gaussian Mixture Model (GMM) and Support Vector Machine (SVM), respectively.

Special attention has to be typically dedicated to good *session compensation*. Simple speaker models without this stage take all the signal variability to create speaker models, which is obviously wrong. The session compensation methods are designed to filter out the unwanted *session variability* and to keep the *speaker variability* that we can use to

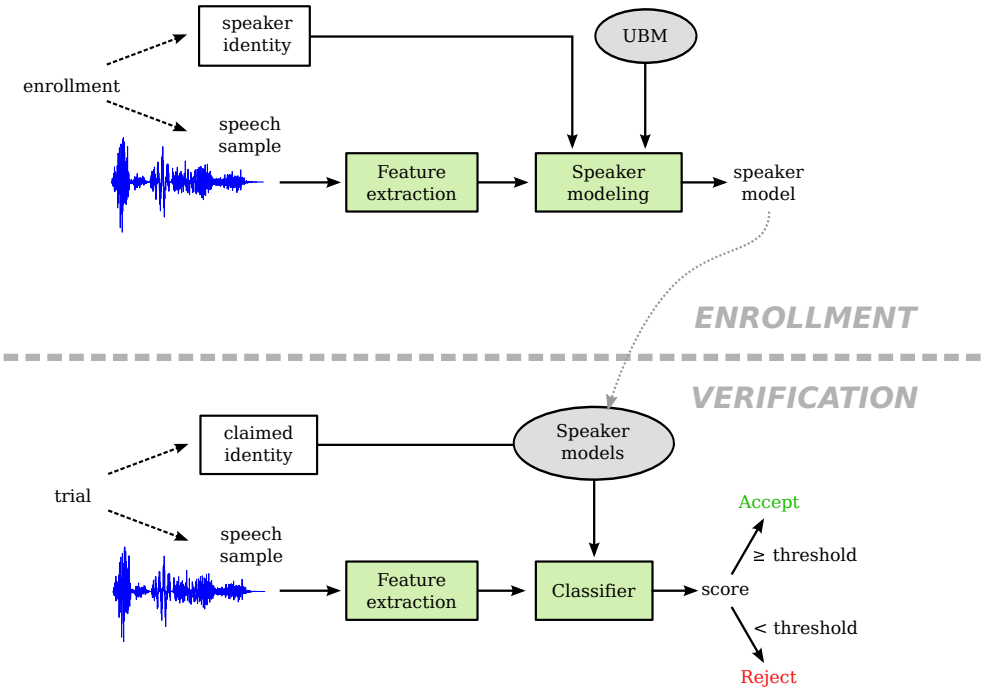


Figure 1.2: Block diagram of general automatic speaker verification system. There are two phases depicted – enrollment of target speakers and verification.

distinguish between speakers. By the term *session variability* we mean any variation in signal characteristics caused by different conditions (mismatch) between the recordings of the same speaker, for example:

- different type of microphone and transmission channel – this is jointly called *channel mismatch* and it is the most important variability to compensate for.
- ambient noise and different room acoustics.
- different phonetic content (in the case of text-independent ASV).
- intra-speaker variability – caused by variations of speaker voice as time passes. This can be short-term, e.g. different speaker mood or variation caused by illnesses, or long-term deviation influenced mainly by speaker aging.

Because both speaker and session variability share a common feature space, they affect each other and thus a mismatch in session conditions can confuse a speaker recognition system based on features from this common space. The ideal case would be to find a feature representation space only affected by speaker variability. Then it would be easy and straightforward to model this distribution and make optimal decisions (in a Bayesian sense). This is the role of session compensation.

1.2.3 Scoring and decision making

At the end of the processing chain lies scoring and decision making module. This module accepts a *trial* as an input. A trial is a pair of test utterance and claimed identity. This module takes the utterance, extracts the features from it and compares them to the saved speaker model. Typically, likelihood ratio for the following two hypotheses is computed [11]:

- H_{target} – The test segment was generated by this speaker-specific model,
- $H_{nontarget}$ – The test segment was generated by a speaker-independent background model.

The bigger the number, the more probable is the first hypothesis. This is also called a soft decision. To make a hard decision, this score has to be thresholded by some fixed value. We give more details on this later in this chapter when discussing how to evaluate recognition performance.

1.2.4 Running a speaker recognizer

During the lifetime of classical ASV system, we recognize three phases:

- **System building and training** – This is the first phase. It typically involves training a background model and running the whole recognition process on some supervised *development* database. The system parameters are tuned to the best performance with respect to the target application needs. From a practical point of view, the processing toolkit and development data have to be selected/created.
- **Enrollment of target speakers** – In this phase, the target speakers to be later recognized are registered to the system using their enrollment data. The speaker model is internally created and stored in the system. This is also often termed as creating each speaker's *voiceprint*.
- **Verification** – An unknown utterance to be verified together with claimed speaker identity is presented to system and the system should correctly answer the question, if the utterance was really pronounced by this speaker or not.

Nowadays, systems often use a so called symmetric scoring. In this case, there is no difference between enrollment and test data, as the trial is formed by two utterances and the system has to decide, whether the two were spoken by the same person or not. The two hypotheses are then [11]:

- H_1 – Both utterances were spoken by the same (one) speaker,
- H_2 – Each utterance was spoken by a different speaker.

In fact, no speaker model is created, because both utterances are handled in the same way – they are transformed to some space, where we can use directly some similarity measure. In such case, the phases of enrollment and verification are merged together.

1.3 Evaluation metrics

In this section, we describe standard measures and tools commonly used to evaluate speaker recognition system performance. Those are: Equal error rate (EER), Detection cost function (C_{Det}) and Detection error tradeoff (DET) curve.

The importance of common evaluation database should be noted here. Direct comparison of evaluation metrics is only possible for the same database, as the datasets differ in difficulty. For a long time, this common evaluation data is governed by the NIST in the scope of Speaker Recognition Evaluation (SRE) campaign[1].

Speaker verification is in general a binary classification task with four possible outcomes [28]:

Table 1.1: Possible outcomes for speaker verification

	Client	**Impostor**
Impostor predicted	False rejection (FR)	True rejection
Client predicted	True acceptance	False acceptance (FA)

Sometimes, false rejection is referred as a *miss* or *nondetection* and false acceptance as a *false alarm*. Probabilities of these events are defined:

$$P_{miss} = \frac{N_{miss}}{N_{tar}}, P_{fa} = \frac{N_{fa}}{N_{non}},$$

where N_{miss}, N_{tar}, N_{fa} and N_{non} are number of false rejections, number of target trials, number of false acceptances and number of false trials, respectively.

P_{fa} and P_{miss} are dependent on each other and the trade-off between them is called *operating point of system* and can be set by altering system's detection threshold. Commonly used fixed operating point is called **Equal error rate (EER)**, and is defined as a point, where $P_{miss} = P_{fa}$.

EER treats the two kind of errors equally, but P_{fa} and P_{miss} are usually not equally expensive. For example, in access applications, false acceptances are much more expensive than false rejections. Thus, operating point sits typically in the area of low false acceptances. This is taken into account in the **detection cost function** C_{Det} used in NIST speaker recognition evaluations. This function sets the operating point with respect to target detection prior probability and costs for the two errors:

$$C_{Det} = C_{Miss} \times P_{Miss|Target} \times P_{Target} + C_{FalseAlarm} \times P_{FalseAlarm|NonTarget} \times (1 - P_{Target})$$

The parameters of this cost function are the relative costs of detection errors, C_{Miss} and $C_{FalseAlarm}$, and the a priori probability of the specified target speaker, P_{Target} [34]. The values for these parameters are summarized in Table 1.2. Note, that recent NIST SRE evaluations use a different cost model. In this work, we do not use such recent evaluation

[1]National Institute of Technology Speaker Recognition Evaluations (NIST SRE) are held regularly to drive the technology forward and measure the state-of-the-art of text-independent speaker recognition, http://www.itl.nist.gov/iad/mig/tests/spk/

Table 1.2: Parameter values for detection cost function used for NIST SRE evaluations (prior to 2012).

C_{Miss}	$C_{FalseAlarm}$	P_{Target}
10	1	0.01

data, so we use the old cost model. To have more interpretable numbers, it is common technique to normalize C_{Det}. As a normalization constant, performance of *default* system is used. Default system has no discriminative power, and its C_{Det} is constant for given relative costs and target prior:

$$C_{default} = \min \begin{cases} C_{Miss} \times P_{Target}, \\ C_{FalseAlarm} \times (1 - P_{Target}) \end{cases}$$

$$C_{Norm} = \frac{C_{Det}}{C_{default}}$$

The symbol C_{Det} is often used for normalized cost function in place of C_{Norm}. We use this practice also – by C_{Det} we automatically mean the normalized version of detection cost function later in our text.

The C_{Det} metric is used to measure the performance of the system given hard decisions. If we have access to true labels for the evaluation data, we use optimized metric **minimum detection cost** C_{Det}^{min}, that minimizes the cost function by setting optimal threshold on scores.

To visualize overall system behavior, the **Detection Error Tradeoff (DET)** curve [28] can be used. It shows system performance for all operating points in terms of both P_{Miss} and $P_{FalseAlarm}$. Both axes are logarithmic. An example of such curve can be seen in Figure 5.1.

Later we will use the **Receiver Operating Characteristic Convex Hull (ROCCH)** curve. This is obtained by interpolating between the discrete points of the ROC curve, which is basically the same as DET curve, but plotted using linear axes. This curve is used to compute a well defined EER from an otherwise „steppy“ ROC/DET curves [9]. We use the ROCCH curve as a smoothed version of the DET curve.

1.4 Score fusion

As we mentioned in previous sections, there are multiple kinds of speech features. We can combine these complementary sources of discriminative information on feature level by stacking all possible feature vectors. This is called feature-level fusion, but it can not be always used, as the features can be very different in nature and with different frame rates. Instead, several separate classifiers are trained side by side and the outputs of these subsystems are merged on score level – score-level fusion. Usually, final decision of state-of-the-art speaker recognition system is a combination of many subsystems. The same approach is also used when combining several biometric modalities, like iris patterns, fingerprints and speaker recognition in security applications.

Each subsystem outputs a score when given a test trial. By convention, higher score values favor true target hypothesis and lower values the opposite. Scores from several

subsytems are then combined using weighted sum [12]:

$$score_f = s(x, \boldsymbol{w}) = w_0 + \sum_{n=1}^{N} w_i s_i(x), \tag{1.1}$$

where $score_f$ is fused output score, N is the number of subsystems and $\boldsymbol{w}$ are the fusion weights. These weights are used to reflect the fact, that some subsystems are weaker than others. They are found via logistic regression wrt. some objective function. This is usually convex function and local minimum can be found easily, for example with conjugate gradient methods [12].

Chapter 2

Feature extraction

In this chapter, we give overview of feature extraction techniques currently used for speech processing tasks. We concentrate on features, that incorporate a temporal context, as it is the main topic of this work.

There are many levels of information available in speech waveform and there is no universal feature extraction technique, so each speech processing task uses its own best-suiting subset of these features. Strictly speaking, different features carry different complementary information, so the best systems are made up of many fused subsystems, each of them utilizing diverse features.

2.1 Information available in speech

From the point of view of speaker recognition, [26] gives the following list of different features available in speech signal.

- **Short-term spectral and voice source features** – spectrum, glottal pulse features. These features are sometimes called „acoustic“ as they describe the voice characteristics corresponding to given vocal tract state. Features are computed from 20 ms long signal segments. It is assumed, that for such time the signal remains stationary and the spectrum can be accurately estimated. The most often used features from this category are Mel-Frequency Cepstral Coefficients (MFCC) and Linear Prediction Cepstral Coefficients (LPCC). These features are very popular, because they are easy to extract and contain most of the discriminative information [27] (based on a fact, that different people have different vocal tract characteristics).

- **Spectro-temporal and prosodic features** – pitch, energy, duration, rhythm, temporal features. Prosodic features (or prosodic patterns) describe speech in suprasegmental regions. Time context in several hundreds of milliseconds is taken into account. Prosodic features of speech are considered as a learned habit and include intonation patterns, speaking rate, pause durations etc. The most important prosodic parameter is the fundamental frequency (or F0) [26] and how it evolves in time. These features are more robust to channel effects, as they do not work directly with signal spectrum.

- **High-level features** - phones, idiolect (personal lexicon), semantics, accent, pronunciation. Extraction and modeling of such features is the most complex one. The

speech recognition backend is needed and also a lot of training data to well estimate speaker models.

In this work, we put special attention on features with temporal context, so we are not going to describe the details of low-level (acoustic) and high-level features in this text. More information can be found in [26] or any literature about speech processing [3]. Note, that feature extraction is in development on all levels. New features like Multitaper MFCC or Spectral Centroid Magnitude and Frequency are being investigated and incorporated to todays speaker recognition systems with benefit of greater robustness. Recent overview of the development can be found in [21].

Besides the above-mentioned *speaker-dependent variability*, there is also unwanted variability caused by several reasons, mainly by noise and channel effects. ASV system must take this into account either at feature extraction stage (use features less affected by such variability) or at the following stages by different kinds of normalization and compensation techniques.

2.2 Application specific temporal features

Speaker recognition domain

Many temporal feature extraction techniques were proposed to be used in the domain of speaker recognition. Except already mentioned delta coefficients, that are somewhere in between the acoustic and temporal paradigm, we can mention the Temporal DCT (TDCT) [27][26], Temporal Patterns (TRAP) [20][5], Modulation Frequency [25] and Time-Frequency Principal Components (TFPC) [29]. Higher level of temporal features is modeling prosodic feature trajectories, like features based on Nonuniform Extraction Region Features (NERF) [22].

Language identification domain

In the domain of language identification (LID), there are also several levels of available information and two different main approaches – acoustic and phonotactic. The later involves phoneme recognition followed by a comparison with a language model. Although the phonotactic approach gives better results, it was shown, that incorporating temporal information from context of around 200 ms greatly reduces the difference in performances of the two paradigms [40]. The Shifted Delta Coefficients (SDC) are typically used for this purpose.

Speech recognition domain

The systems for automatic speech recognition (ASR) usually use the features covering longer time span because it is more convenient for phone recognition. The TRAP feature or more recent bottleneck features [19] are commonly used for phoneme recognition. These features are extracted from several stacked acoustic features by a neural network. It was experimentally shown, that the best context size for such features is about 310 ms [32].

2.3 Selected temporal feature extraction methods

In this section, we give a list of feature extraction methods, that make use of temporal context. This added information is known to contain another speaker specific information, complementary to acoustic features. So even if the acoustic features carry the greatest portion of speaker discriminative information [21], temporal features are of great importance.

Delta and acceleration coefficients

For a long time [38], most of the acoustic ASV systems have been making use of the local temporal derivative estimates to capture local speech dynamics. Adding time derivatives to the cepstral feature vector improves recognition performance [42].

Computing the Δ and $\Delta\Delta$ coefficients is commonly implemented as a least-square approximation of the local slope and calculated over multiple frames (typically window of 5-9 frames [2]).

The regression formula for computing delta coefficient Δ_i for a feature C_i is given by [42][2]:

$$\Delta_i(t) = \frac{\sum_{k=-N}^{N} kC_i(t+k)}{\sum_{k=-N}^{N} k^2} = \frac{\sum_{k=1}^{N} k[C_i(t+k) - C_i(t-k)]}{2\sum_{k=1}^{N} k^2}, \tag{2.1}$$

where N determines size of the window across which the regression is computed.

Often, also the second order derivatives are computed. These are called double-delta coefficients or acceleration coefficients. They are computed using the same formula from the first order coefficients. Special handling is needed at the beginning and end of signal.

Shifted Delta Cepstral coefficients (SDC)

Described in [40][14], Shifted Delta Cepstral features are a way, how to incorporate even longer temporal context. Although these features are mainly used for LID, there were some attempts [14] to use it for speaker verification.

In principle, the SDC feature vector is constructed by stacking the delta features sampled from context of about 200 ms.

The SDC feature extraction is parametrized by 4 coefficients [14]:

- N: number of cepstral coefficients (typically 12)
- d: spread of the delta computation
- P: gaps between successive delta computations
- k: number of delta vectors for concatenation

For given time t, the difference vectors are computed:

$$\Delta\boldsymbol{c}(t,i) = \boldsymbol{c}(t+iP+d) - \boldsymbol{c}(t+iP-d)$$

and the final SDC feature vector is then formed by stacking k of these differences:

$$SDC(t) = [\Delta\boldsymbol{c}(t,0)^t \quad \Delta\boldsymbol{c}(t,1)^t \quad \ldots \quad \Delta\boldsymbol{c}(t,k-1)^t]^t \tag{2.2}$$

Many combinations of these parameters are possible, we found usage of N-d-P-k = 7-1-3-7 [14] , 10-1-1-3 [40] and many others. For example, with the 7-1-3-7 configuration, the features have dimensionality of 49 and timespan of 220 ms (for frame length 20 ms and time shift of 10 ms [27]).

It was shown [40], that GMM-based LID systems utilizing SDC features can perform as well as phonetic LID systems with greatly reduced computational cost.

Temporal DCT (TDCT)

Temporal Discrete Cosine Transform [27][26] extracts speech dynamics from the same space as SDC does – from the trajectories of cepstral coefficients. Compared to SDC, it employs different way to reduce dimensionality of final features. Instead of sampling this space (see SDC parameter P), the DCT transform is applied to a vector of consecutive cepstral coefficients and only the low-frequency components are retained and then stacked together from all filterbank bands. That is because low-frequency components contain most of the energy.

TDCT features are extracted for each frame. The use of DCT rather than DFT magnitude retains also relative phase, so it can preserve both phonetic and speaker-specific information. Improvement over the cepstral systems by fusing the match scores of the cepstral and temporal features is rather modest and more research is required [26].

Results in [27] observed on the NIST 2001 corpus indicate, that SDC and TDCT features perform similarly and outperform the MFCC+Δ+Δ front-end.

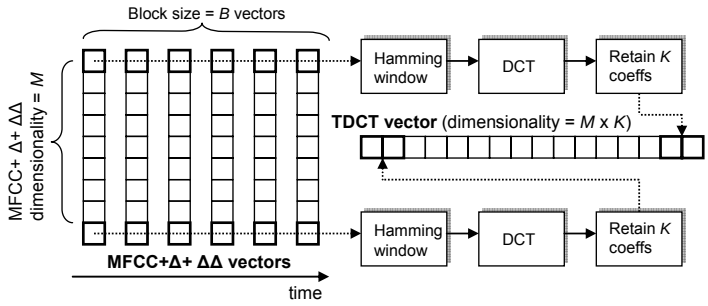


Figure 2.1: Illustration of the TDCT feature computation. Image taken from [27].

Temporal Patterns

The TempoRAl PatternS (TRAPS), proposed in [20] for phoneme recognition, later investigated also for ASV [5] is yet another approach, how to extract temporal context from frequency band trajectories.

Extraction procedure for ASV described in [5] is very similar to the TDCT one. Time trajectories of 18 Mel-filter banks (150 ms in length) are processed in parallel firstly by applying Hamming window and then computing Discrete Fourier Transform (DFT). The magnitudes are then filtered by cancelling all frequencies except the interval 1 to 16 Hz. The final feature vector is obtained by stacking these outputs and applying dimensionality reduction procedure – either LDA or PCA – to squeeze highly dimensional space to 24 dimensions.

In [5] they compare the performance of TRAPS with classical acoustic MFCC+Δ/GMM approach and some results are given. No information is given about possible fusion of these features, as this could be in our opinion more beneficial than using either cepstral features or temporal ones. They also mention that using TRAPS is beneficial especially for noisy conditions.

Modulation Frequency

To extract features like speaking rate, similar approach to TDCT was proposed in [25], called *Joint acoustic-modulation frequency*. This method extracts frequency content of sub-band amplitude envelopes.

The final feature vector is obtained per-utterance by averaging frame-wise feature vectors computed over 300 ms window. Only modulation frequencies $0-20$ Hz are kept, as they are linguistically and perceptually the most relevant modulation frequencies of speech [25].

The difference from TDCT is that trajectory processing is computed over spectral sub-bands instead of cepstral sub-bands using DFT rather than DCT. Also, local information for the whole utterance is averaged to form one feature vector per utterance.

Time-Frequency Principal Components

We end this (incomplete) list of feature extraction techniques with Time-Frequency Principal Components (TFPC) [29] as it can be seen as a generalization of many of above-mentioned techniques. Authors of this method developed a formalism called *filtering of spectral trajectories* that allows to describe several feature extraction techniques like cepstral analysis or Δ coefficients computation by means of matrix multiplication (note, that multiplication in spectral domain is actually convolution/filtering in time domain).

Common characteristic of above-mentioned feature extraction methods is that they process each sub-band trajectory independently (i.e. component by component). The TFPC instead takes the spectro-temporal matrix as a whole, counting with possible inter-component correlations. The dimensionality reduction is accomplished by PCA, thus the name Time-Frequency Principal Components.

2.4 Feature normalization

The objective of feature normalization is to compensate for the effects of session mismatch and to enable more effective modeling of speaker differences by scaling or warping the feature vector [1]. Some channel compensation is convenient on feature level for two reasons: Firstly, many channel effects can be handled, because of their nature. For example, convolutional noise can be by reduced by Cepstral Mean Subtraction, because this noise becomes additive in frequency domain and so centering features effectively „normalizes“ the utterances. Secondly, such signal enhancement is independent of speaker modeling stage, so it is applicable to any speaker recognition system.

Most of the conventional feature normalization techniques are normally applied in the cepstral domain, as a post-processing of extracted MFCC features [1]. This is depicted in Figure 2.2.

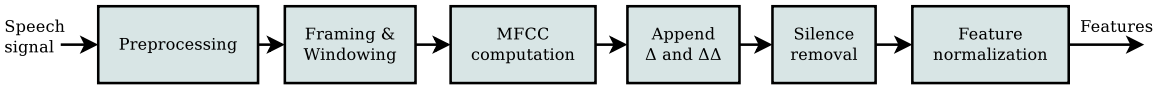


Figure 2.2: Feature extraction chain of a typical acoustic ASV system.

Cepstral Mean and Variance Normalization (CMVN)

Sometimes also referred as Cepstral Mean Subtraction (CMS), CMVN [17][1] is an efficient method to remove mismatch between recordings caused by convolutive channel noise. In the log-spectral and cepstral domains, convolutive channel noise becomes additive [26] and so subtraction of the mean cancels out such mismatch. In [17], the authors note, that it is also effective in reducing long-term intra-speaker spectral variability.

Variance normalization is performed to equalize feature variances by dividing each feature by its standard deviation.

Although convolutive channel noise is in most cases constant for the whole utterance, often sliding-window variant of CMVN, Short-Time Mean and Variance Normalization (STMVN) is used. This is motivated by real-time applications, where the whole utterance is not available ahead and by the fact, that the mean can change in time. In [1], the authors compare different feature normalization methods including STMVN concluding that STMVN, with an i-vector system, provides comparable speaker verification results to that of Short-Time Gaussianization (described later).

Feature Warping and Short-Time Gaussianization

Feature Warping [35] and Short-Time Gaussianization [41] are similar feature normalization techniques. The normalization is done by „warping“ the cumulative feature distribution to follow a reference distribution (simply a Gaussian). This should result in cepstral feature representation more robust to additive noise and linear channel effects. Histogram equalization of pixel intensities is a similar technique known from image processing.

The difference between the two is in a preprocessing by global linear transformation in the case of STG, used to decorrelate the features.

It was shown [41], that STG outperforms Feature Warping. On the other hand, it is more complex to implement [26].

Chapter 3

Speaker modeling and session compensation

This chapter gives an introduction to generative speaker modeling and session compensation techniques. A nontrivial math is needed to compensate for the different kinds of variabilities in speech signal that cause significant problems for simple speaker models.

Prior to subspace models, the speaker modeling was performed by simply MAP adapting the world background model [37] using speaker enrollment data, which does good job for clean „laboratory“ conditions, but it does not perform well on mismatched data. This mismatch often arises in real world applications and so more sophisticated models were proposed.

Session variability compensation can be applied on different levels of speaker recognition system. There are ways to compensate for channels effects in feature space, to compensate for session variability in model space and often, some sort of score normalization is also needed to mitigate the scoring offset arising from session mismatch. In this chapter, we will describe model level session compensation, as the most complex one. We start by describing a basic UBM-GMM model, to introduce the main principles and symbols.

3.1 Gaussian Mixture Model

To model multimodal data, like different kinds of speech features, the Gaussian Mixture Model (GMM) can be used. GMM is defined as a weighted sum of Gaussian functions [4]:

$$p(\boldsymbol{x}) = \sum_{c=1}^{C} \pi_c \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{3.1}$$

$$\sum_{c=1}^{C} \pi_c = 1, \tag{3.2}$$

where π_i is a weight for a Gaussian i ($0 \leq \pi_i \leq 1$). Together with mean vector $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$, those are the parameters of GMM, that has to be estimated on training data using iterative algorithm, like EM (Expectation Maximization) algorithm. The covariance matrix $\boldsymbol{\Sigma}_i$ is often restricted to be diagonal, especially for high-dimensional data. Parameter C, the number of mixture components, has to be chosen with respect to the data distribution.

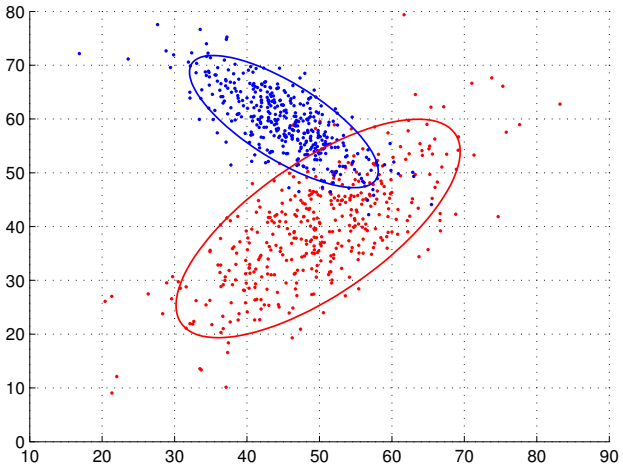


Figure 3.1: Two-dimensional example of a mixture of two Gaussian distributions. The ellipses show component contours. In real data, the assignment of data points to mixture components is not given and that is why we must use EM iterative algorithm to fit the GMM onto multimodal data.

Given some GMM with parameters $\boldsymbol{\Theta} = \{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, we can compute the likelihood of data $\boldsymbol{x}$, given this model. To compute the likelihood on speech utterance $X = \{\boldsymbol{x}_1, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, the average computed over all the frames is used. For practical reasons, the log of the likelihood function is taken:

$$\log p(X|\boldsymbol{\Theta}) = \frac{1}{N} \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{\Theta}). \tag{3.3}$$

Typically, the GMM for acoustic speaker recognition purposes contains 512, 1024 or 2048 components.

3.1.1 Background model and MAP adaptation

It was shown [39], that it is convenient to adapt specific models from one „mean model“, called UBM (Universal Background Model), trained with a big amount of exemplary data. With this approach, specific models are more robust and we can perform the adaption even with a limited amount of target data. The adaptation is also quicker than training a new GMM iteratively. There are several adaptation techniques, like MAP (Maximum A Posteriori) or MLLR (Maximum Likelihood Linear Regression) adaptation.

For the MAP adaptation, the model parameters are shifted towards the target data, depending on their occupation probabilities. This is important, because we modify only the parameters, for which there are some target data examples, leaving the parameters without adaptation data untouched. For the task of speaker recognition, it suffices to adapt only the means of UBM mixtures. The whole process is explained in detail in [39].

3.2 Subspace modeling

When considering a typical GMM with 512 components and 60 dimensional features, the number of parameters of such model is very large. This has many drawbacks like the curse of dimensionality problem, impossibility to use some standard machine learning methods and big computing costs. The subspace methods are based on an assumption, that all the parameters lies in fact in a parameter subspace (i.e. there is some latent variable). Moreover, using subspaces, we can separately model the speaker and session variability. All these methods work with the term supervector, so we start by introducing this.

3.2.1 Concept of supervectors

When doing adaptation from UBM, we can adapt mixture weights, means and variances. As discussed in [39], for speaker recognition task, just the means can be adapted with no accuracy lost. So the speaker model if fully described only by means of all mixture components. For a GMM with C components and input feature vector of size F, we can concatenate individual mean vectors together to obtain a vector of dimension CF, which is called supervector [26].

Thus, using this approach, we transformed the variable sized speaker representation in the form of sample speech data into fixed length supervector. Fixed length representation allows us to use directly general pattern matching algorithms and classifiers, for example discriminative classifier Support Vector Machines (SVM).

To compensate for channel effects in supervector space, different methods were proposed based on the same idea, that the total variability can be decomposed into speaker variability and session variability components. The problem of finding and modeling corresponding subspaces is analyzed further in the following text.

3.2.2 Joint Factor Analysis of speaker and session variability

In the Joint Factor Analysis (JFA) [23] model, the speaker and channel variability are modelled separately by two low-dimensional latent variables. The supervector space is assumed as a linear combination of these latent variables. The supervector space can be thought as sum of orthogonal speaker dependent and channel dependent components $\boldsymbol{s}$ and $\boldsymbol{c}$:

$$\boldsymbol{M} = \boldsymbol{s} + \boldsymbol{c} \tag{3.4}$$

These components can be expressed in terms of latent variables:

$$\boldsymbol{s} = \boldsymbol{m} + \boldsymbol{V}\boldsymbol{y} + \boldsymbol{D}\boldsymbol{z} \tag{3.5}$$

$$\boldsymbol{c} = \boldsymbol{U}\boldsymbol{x}, \tag{3.6}$$

where $\boldsymbol{m}$ is now channel and speaker independent mean, $\boldsymbol{V}$ is a rectangular matrix of low rank that defines a speaker subspace, $\boldsymbol{D}$ is $CF \times CF$ diagonal residual matrix which captures the residual variability, $\boldsymbol{U}$ is a rectangular matrix of low rank that defines a session subspace and $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ are normal distributed random vectors.

The complete model is then:

$$\boldsymbol{M} = \boldsymbol{m} + \boldsymbol{V}\boldsymbol{y} + \boldsymbol{D}\boldsymbol{z} + \boldsymbol{U}\boldsymbol{x}. \tag{3.7}$$

Training of this model is not an easy task and we refer to the original paper [23]. Scoring is done by computing the likelihood of the test utterance feature vectors against a session-compensated speaker model ($\boldsymbol{m} - \boldsymbol{U}\boldsymbol{x}$).

3.2.3 Total variability subspace and i-vectors

During the summer 2008 JHU workshop on Robust Speaker Recognition it was shown, that the JFA does not fully separate the speaker and session subspaces, as using the channel factors has fairly big discriminative power [18]. After that, it was proposed in [15] to merge the two subspaces and perform channel compensation on the total factors, called i-vectors.

The i-vector approach has become a popular technique for speaker recognition with later applications also for language identification [16][30] and other speech related tasks. This technique offers a way to transform high dimensional sequential data into low dimensional (order of hundreds of dimensions) fixed length vectors, called i-vectors.

The generative factor analysis model is used to model the supervector variability in a low-dimensional subspace. The total variability subspace model is defined by:

$$\boldsymbol{\mu} = \boldsymbol{m} + \boldsymbol{T}\boldsymbol{w}, \tag{3.8}$$

where the factor $\boldsymbol{w}$ is commonly referred as the i-vector, $\boldsymbol{m}$ is the channel and speaker independent mean supervector (simply the UBM) and $\boldsymbol{T}$ is a matrix of bases spanning the subspace with important speaker and channel variability. We assume, that $\boldsymbol{w}$ is distributed normally among $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

This technique reduces the number of parameters of the speaker model. Instead of having CF variables in case of full supervector space, now only components of $\boldsymbol{w}$ have to be estimated. This is important especially when there is not enough training data available. Moreover, compared to the JFA processing needs, the extraction of i-vectors is much faster.

The analytical solution for this model does not exist and the model parameters has to be estimated using iterative algorithm. A possible approach to estimate the i-vectors can be found in [31].

With this technique, the variable length session is transformed into compact fixed-length and low-dimensional representation and it can be seen just as an input data re-parametrization.

Because i-vectors still contain all the variability, session compensation has to be performed on this level. Because of the small size of i-vectors, this can be very quick process. The simplified version of JFA is commonly used for this purpose, called Probabilistic Linear Discriminant Analysis (PLDA) [36][13].

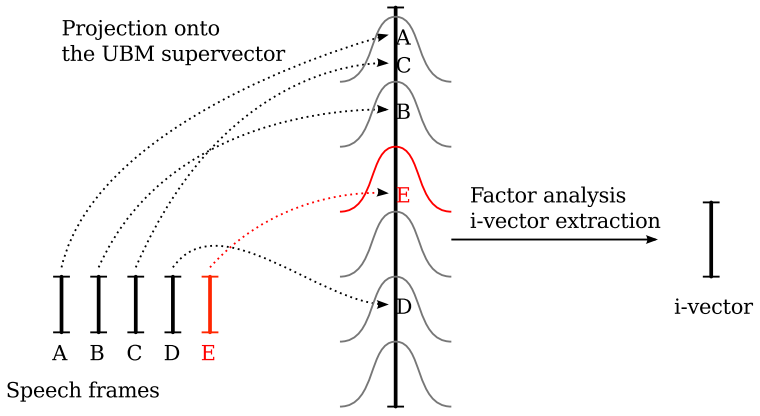


Figure 3.2: Simplified diagram showing the i-vector method of transformation variable length feature vector into fixed length i-vector.

3.2.4 I-vector space scoring and session compensation

There are currently two main scoring techniques in i-vector space. Note, that when using i-vectors, symmetric scoring is used, i.e. there is no difference between training and test. The question is stated as: do these utterances come from the same speaker or not?

The first method uses cosine similarity scoring with session compensation performed by Linear Discriminant Analysis (LDA) followed by Within-Class Covariance Normalization (WCCN). The second, probabilistic approach, is called Probabilistic Linear Discriminant Analysis (PLDA).

Cosine distance scoring

Proposed in [15], cosine distance is an efficient method how to compare two i-vectors. Defined as:

$$score_{cosine}(\boldsymbol{w}_1, \boldsymbol{w}_2) = \frac{\langle \boldsymbol{w}_1, \boldsymbol{w}_2 \rangle}{||\boldsymbol{w}_1||\ ||\boldsymbol{w}_2||}, \tag{3.9}$$

it effectively measures the angle between the two vectors. This score can be used directly as a soft decision. Because i-vectors still contain the session information, the session compensation has to be performed before this step. This is usually done by projecting the i-vectors using LDA followed by WCCN [15][18].

Probabilistic Linear Discriminant Analysis

The Probabilistic Linear Discriminant Analysis (PLDA) [36] was originally developed for face recognition purposes and then adopted by speaker recognition community. In face recognition, there is also useful variability caused by differences in faces of different persons and nuisance variability caused by different lightning conditions and pose. The relation between PLDA and LDA is analogous to the relation of Factor analysis and PCA. It is a generative model, where i-vector is modelled as [18]:

$$\boldsymbol{w} = \boldsymbol{\mu} + \boldsymbol{V}\boldsymbol{y} + \boldsymbol{U}\boldsymbol{x} + \boldsymbol{\epsilon}, \tag{3.10}$$

where the terms $\boldsymbol{\mu}$, $\boldsymbol{V}\boldsymbol{y}$ and $\boldsymbol{U}\boldsymbol{x}$ have the similar meaning as for JFA and the $\boldsymbol{\epsilon}$ represents residual data noise with diagonal covariance $\boldsymbol{\Sigma}$. The latent variables are assumed to be Gaussian with zero mean and unit variance.

When given two test i-vectors, trained PLDA model can be used to evaluate the likelihood ratio score:

$$score_{PLDA}(\boldsymbol{w_1}, \boldsymbol{w_2}) = \log \frac{p(\boldsymbol{w_1}, \boldsymbol{w_2}|H_1)}{p(\boldsymbol{w_1}, \boldsymbol{w_2}|H_1)}. \tag{3.11}$$

Both nominator and denominator can be evaluated analytically. The equations for numerator and denominator are, respectively [18]:

$$\mathcal{N}\left(\begin{bmatrix}\boldsymbol{w_1}\\ \boldsymbol{w_2}\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\mu}\end{bmatrix}, \begin{bmatrix}\boldsymbol{V}\boldsymbol{V}^{\boldsymbol{t}} + \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{t}} + \boldsymbol{\Sigma} & \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{t}}\\ \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{t}} & \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{t}} + \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{t}} + \boldsymbol{\Sigma}\end{bmatrix}\right) \tag{3.12}$$

and

$$\mathcal{N}\left(\begin{bmatrix}\boldsymbol{w_1}\\ \boldsymbol{w_2}\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\mu}\end{bmatrix}, \begin{bmatrix}\boldsymbol{V}\boldsymbol{V}^{\boldsymbol{t}} + \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{t}} + \boldsymbol{\Sigma} & 0\\ 0 & \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{t}} + \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{t}} + \boldsymbol{\Sigma}\end{bmatrix}\right). \tag{3.13}$$

It was shown [24], that using heavy-tailed prior distribution like student's t-distribution increases performance and thus the assumption of i-vector gaussianity was invalidated.

Later, it was shown [8], that length normalization can be used to effectively gaussianize the i-vectors to be used with gaussian-prior PLDA. Length normalization is defined as:

$$\boldsymbol{w'} = \frac{\boldsymbol{w}}{||\boldsymbol{w}||} = \frac{\boldsymbol{w}}{\sqrt{\boldsymbol{w}^t\boldsymbol{w}}}. \tag{3.14}$$

The two-covariance model

There is a simplified version of PLDA called *two-covariance model* [10] which we use in our work, where the model is decomposed into:

$$\boldsymbol{w} = \boldsymbol{y_s} + \boldsymbol{\epsilon}, \tag{3.15}$$

and the priors are defined as:

$$P(\boldsymbol{y_s}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{B}), \tag{3.16}$$

$$P(\boldsymbol{w}|\boldsymbol{y_s}) = \mathcal{N}(\boldsymbol{y_s}, \boldsymbol{W}). \tag{3.17}$$

Since (3.16) is conjugate prior to (3.17), the posterior probability of this model can be directly evaluated, see [10]. The matrices $\boldsymbol{B}$ and $\boldsymbol{W}$ are between-speaker and within-speaker covariance matrices:

$$\boldsymbol{B} = \sum_{s=1}^{S} \frac{n_s}{n}(\boldsymbol{y}_s - \boldsymbol{\mu})(\boldsymbol{y}_s - \boldsymbol{\mu})^t \tag{3.18}$$

$$\boldsymbol{W} = \frac{1}{n}\sum_{s=1}^{S}\sum_{i=1}^{n_s}(\boldsymbol{w}_i^s - \boldsymbol{y}_s)(\boldsymbol{w}_i^s - \boldsymbol{y}_s)^t, \tag{3.19}$$

where n_s is the number of utterances for speaker s, n is the total number of utterances, $\boldsymbol{w}_i^s$ are the i-vectors of sessions of speaker s, $\boldsymbol{y}_s$ is the mean of all the i-vectors of speaker s and $\boldsymbol{\mu}$ represents the overall mean of the training dataset [7].

The difference between general PLDA is, that covariance matrices are full-rank, as opposed to possible subspaces in general PLDA model [18].

Chapter 4

Capturing temporal context using i-vector paradigm

Here we present a possible application of the i-vector paradigm to the task of temporal feature extraction. We start by a discussion of interesting aspects of the i-vector paradigm that led us into this.

4.1 Motivation

We think, that the concept of i-vectors (i.e. the factor analysis applied on the adapted UBM supervector) is a very good information extractor in general. Speech frames are firstly projected into high dimensional space (MAP adaptation of UBM) to form a supervector, that is then reduced by the factor analysis into a vector of small size. From a variable length input it is possible to obtain one vector with hopefully all the relevant information. This elegantly solves the data redundancy problem, which all the spectro-temporal feature extraction methods have to struggle with, because consecutive speech frames have generally a lot of in common.

4.1.1 Possible pitfalls

One problem that could arise when using this technique for temporal feature extraction from short speech segments is, that it does not retain the information of the frame order. The i-vector extracted from speech segment parametrized by frames [**a**, **b**, **c**] will be the same also for the cases [**c**, **b**, **a**] or [**c**, **a**, **b**]. This problem is illustrated in Figure 3.2. So, in the final feature vector, there will be information about frames present in the segment, but without taking into account their relative configuration. In this work we assume, that from a statistical point of view, such sources of possible misclassification are rare in real data.

Normally, for speaker verification task, the i-vector is extracted from the whole utterance. The resulting i-vector then carries information about the utterance as a whole, because local phonetic content is „averaged out“. If we will gradually reduce the length of the utterance, local phonetic content will become more visible. Of course, the data used to train the total variability matrix has to follow the same path. Also, the i-vectors can not be considered anymore as speaker and channel factors only, as the factor loadings matrix is not being trained on variability between utterances, but rather on variability between short segments, that vary a lot with respect to phonetic content.

The common size of the i-vector used to represent the whole utterance is 400. This work tries to answer the question, what is the dimensionality of subspace, where lies the main variability amongst very short segments of speech. Naturally, we want this number to be as low as possible.

4.2 System design

The original idea is depicted in Figure 4.1. After classical signal parametrization (like MFCC or PLP features), the features are re-parametrized by extracting i-vectors over the sliding window. There are two parameters, *window size* and *shift*. The total variability matrix used during i-vector extraction is trained on the variability between many exemplary speech segments of given size from development data. By choosing specific exemplary speech segments to train the total variability matrix (like speech segments aligned to phonemes in our case), we can extract features (i-vectors) with different properties. This process is computationally very demanding and some optimizations or approximations are needed for practical use.

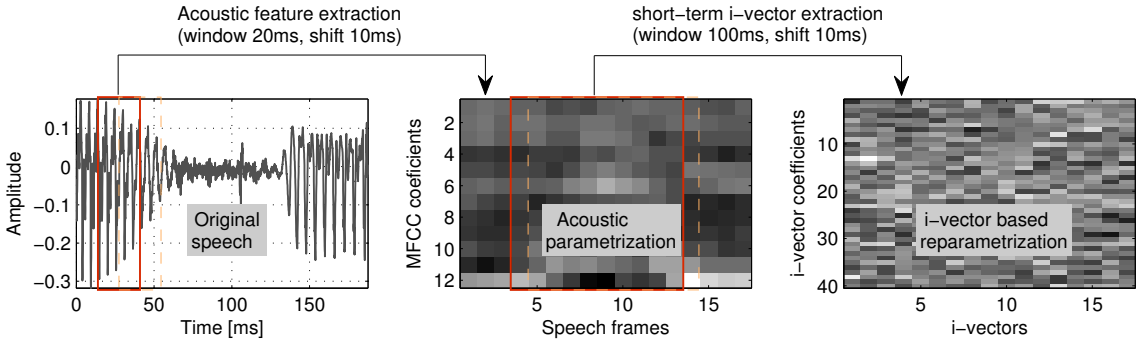


Figure 4.1: Overview of the proposed approach.

4.2.1 Classification and scoring

For a classical GMM-UBM system, the class models are created by MAP adaptation of UBM using enrollment data. The UBM is trained beforehand to represent the overall distribution of features. The scoring of the utterance is then done frame-wise, i.e. if the model parameters are $\boldsymbol{\Theta}$ and testing utterance is $\boldsymbol{X} = \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$, we compute the model log-likelihood given the data as:

$$\mathcal{L}(\boldsymbol{\Theta}|\boldsymbol{X}) = \sum_{i=1}^{N} \frac{1}{N} \log p(\boldsymbol{x}_i|\boldsymbol{\Theta}). \tag{4.1}$$

We can use the same approach also for the re-parametrized features. But in this case, instead of computing the probability of some speech segment as an average probability of its (independent) frames, we use one frame that represents the segment as a whole. Also, the structure of the UBM that is trained on such features is different. In this work, we assume, that the distribution of the re-parametrized features is again multi-modal with similar distribution, that can be again modeled by GMM-UBM.

Chapter 5

Experimental results

This chapter describes the work we have done to verify benefits of the proposed approach. The experiments are divided into two parts.

Firstly, we evaluate our approach on phoneme verification task. As this is a brand new concept in the field of speech feature extraction, we use phoneme verification as a benchmark, to evaluate the characteristics of newly extracted features. We had to do that, as the first experiments on speaker verification task were unexpectedly bad.

After that, we try to apply this approach to speaker verification task, evaluating its performance on the widely used database NIST SRE 2008.

These two evaluations have big differences in their experimental setups and evaluation criteria, so we describe the systems and evaluation databases separately for each set of experiments. We start with the first set of experiments realized on ESTER speech database in the next section.

5.1 Preliminary study of i-vector based feature reparametrization

Here we present the preliminary experiments that we conducted on annotated speech database ESTER after we had obtained very poor results with the original idea described in Section 4.2 on the NIST SRE 2008 speaker recognition evaluation task. The bad results were caused probably by the fact, that we did not know suitable values for system parameters yet. We chose the annotated (i.e. with phoneme segmentation) database to analyze the behavior of i-vector extraction from very short speech segments of defined content. On this database, the contextual phoneme verification task was evaluated, to have more interpretable results and to get more insight into the re-parametrization step. Contextual phonemes are units of the time span that is close to our first experiments (segment size of around 10 frames). The window size used in the following experiments is thus variable, according to the phoneme length.

5.1.1 Contextual phoneme verification task

For this set of experiments, verification task was selected. There are generally many thousands contextual phoneme classes and so the verification task is more practical than classification task.

We used contextual phoneme classes instead of classical single phonemes because the labeling was available for the ESTER database. The contextual phonemes are used mainly because they offer higher resolution that can handle coarticulation effects.

Contextual phoneme is defined using its central phoneme and a left and right neighborhood. We use the notation using colons: $\alpha : pp : \beta$, where α and β are left and right neighboring phonemes and pp is the central phoneme.

Phoneme verification is the process of checking a claimed phoneme label against some test speech segment. Phoneme boundaries are given, so this is only a pattern matching problem. It is very similar and analogous to speaker verification described in introduction. We can use the EER to measure system performance. An example of results file augmented with a key is in Table 5.1.

Table 5.1: Illustrative example of the results file for phoneme verification task.

Test segment	**Claimed class**	**System output**	**Key**
seg01.wav	ss:yy:on	0.46	true
seg02.wav	pp:aa:rr	-2.46	false
seg03.wav	aa:ss:yy	-3.12	false
⋮	⋮	⋮	⋮

5.1.2 ESTER database

The ESTER database contains broadcast recordings in French, that were collected from different radio stations to serve as an evaluation database for campaign of Broadcast News enriched transcription systems using French data. The orthographical transcription and segmentation is available[1].

We used the 2000 hours part of the non transcribed broadcast news shows data from the ESTER with the automatic contextual phoneme transcription available at LIA to create the database subset divided into 11201 parts. Each part was represented by different examples of given contextual phoneme. We limited the amount of examples for each contextual phoneme to maximum of 1000 (randomly selected) examples. We also filtered out the classes of contextual phonemes with less than 30 examples and the class for silence. The reason of limiting the database was to have the balanced number of examples for each class.

This subset contains 11201 contextual phonemes with average length of 8.46 frames and average number of examples for contextual phoneme class 400. There are approximately 100 hours of speech.

This data was used to train the UBM and the total variability matrix. To create the target training and test data (the evaluation data), we chose 90 contextual phoneme classes with 1000 examples from the development data. The 1000 examples were divided randomly into test and training sets. For each contextual phoneme class, 10 examples were used for testing and 990 examples for training (as an adaptation data). To create the trial definition file, for each of the 90 modeled phonemes, the 10 test examples were taken to represent target trials and 100 examples were randomly selected from the test examples of other phonemes to represent non-target trials. Total number of trials is thus 9900. The testing and training data were disjoint, but they were part of the development data used to train the frame-level UBM and the total variability matrix (re-parametrization process).

[1] http://catalog.elra.info/product_info.php?products_id=999

Evaluation data summary:

- number of contextual phoneme models: 90
- adaptation data: 990 examples for each phoneme
- test data: 10 target trials and 100 non-target trials for each phoneme, 9900 trials in total

5.1.3 Contextual phoneme verification using ESTER database

In this set of experiments, we wanted to find reasonable size of i-vectors that we can use for further experiments. Another question was if we loose the information during proposed feature re-parametrization. In theory, the i-vectors are projections of supervectors to low-dimensional space, where the supervectors are MAP adapted versions of the UBM. We have therefore compared the recognition performance of contextual phoneme models created by:

1. MAP adaptation of the UBM means using PLP features – baseline system
2. MAP adaptation of the UBM means on the level of re-parametrized features (an i-vector extracted for each contextual phoneme example from training data) – i-vector system

To train the total variability matrix, we used available examples of different contextual phonemes from development set.

5.1.4 Baseline system: GMM-UBM on acoustic feature level

For a comparison, we firstly evaluate our task using basic GMM-UBM approach. The aim is not to create the best phoneme verification system, but our aim is rather to find, how much information is transferred from frames into adapted supervector and after, in the following experiments, how much we can reduce its size using factor analysis.

The speech signal was parametrized into 39 dimensional frames using PLP features with 13 coefficients and the delta and acceleration coefficients. By using the delta coefficients, we already add some temporal information into features, but because this is de facto standard in acoustic speaker recognition, we kept this practice.

The UBM was trained on pooled data for all contextual phonemes and then specific models were created with MAP adaptation of means, using the training examples for a given contextual phoneme (about 80 seconds of data). Note, that any information of the frame order within the phoneme example (normally modeled by a 3-state HMM) is not used here. The performance in terms of %EER is given in Table 5.2.

Table 5.2: The results of the baseline system (GMM-UBM PLP+Δ+$\Delta\Delta$ features) with different GMM sizes. (%EER)

Number of mixtures	64	128	256	512
GMM-UBM	5.57	5.32	4.57	4.53

For further experiments, we chose to use the number of mixture components to be 128 for practical reasons (speed and amount of available training data) and also because the same number was used in [6] for similar purposes of representing the HMM states. This results in a supervector size of 4992.

5.1.5 i-vector system: GMM-UBM using re-parametrized features

To create phoneme models based on i-vector features, we used firstly the speaker recognition approach, i.e. scoring one i-vector against another one using some metric (Euclidean distance or cosine similarity). This did not bring good results. We think this was caused partly by insufficient amount of training data used to train the total variability matrix and partly by large variation of training segments for each class. We also tried to use the average i-vector to represent each class, without much gain. So we model the intra-class i-vector feature variability explicitly by GMM.

The total variability matrix was trained using all the examples for all the phonemes. After extraction, resulting i-vectors were normalized to have globally zero mean and unit variance, as this condition was not satisfied.

The phoneme models were then created in the same way as for the baseline system (GMM-UBM), except for the difference that we used extracted i-vectors instead of PLP frames as features. The results are shown in Table 5.3.

Table 5.3: Contextual phoneme recognition performance (%EER) for i-vector based system. Both mixture models (i.e. UBM on PLP frames used to extract i-vectors and the secondary mixture model for i-vectors) had 128 components.

TV matrix rank	10	20	30	40	45	50	60	80	100
i-vector GMM	8.44	6.22	5.03	**4.78**	4.89	5.53	**5.11**	5.22	5.12

These results show, in comparison to the baseline system, that for this task it is possible to re-parametrize the speech signal using small i-vectors, which are extracted from segments only a few frames long.

In comparison with the simpler baseline system, the results are slightly better. It is clear, that this is caused by added temporal context.

5.1.6 System combination

We present the last results in Table 5.4. It is the weighted score fusion of the baseline reference system and the i-vector GMM system. As this is a preliminary study, the mixing weights were simply set to be equal. The results of the individual systems are shown also for reference.

Table 5.4: Score level fusion of the baseline system and the i-vector GMM system. If not stated otherwise, the size of used i-vectors was 30.

System	%EER
Baseline GMM (baseline)	5.32
i-vector based GMM (iGMM)	5.03
baseline + iGMM	3.64
baseline + iGMM (rank 60)	3.11

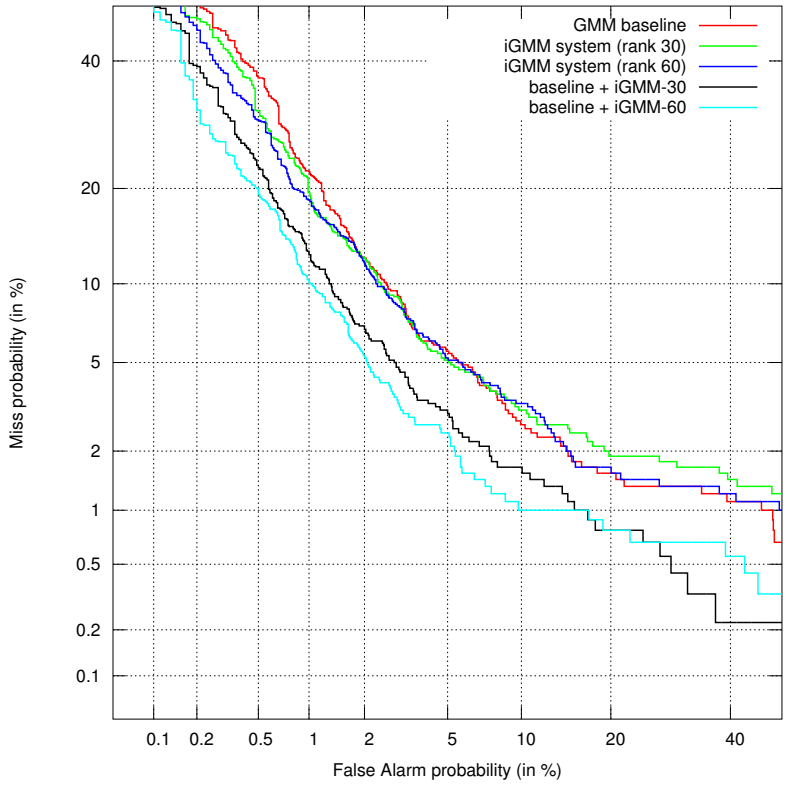


Figure 5.1: DET curves for selected phoneme verification systems and for their score level fusion. By the term iGMM we mean GMM using i-vectors as features.

5.1.7 Discussion

The results obtained using scores fusion show, that our i-vector GMM system offers some complementary information that can be explained as an additional discriminative information extracted from temporal context.

The last line of Table 5.4 presents the results of fusion of baseline system with the i-vector GMM system that uses the i-vectors of size 60. Even if the performance of this individual system (EER=5.11) compared to the best i-vector GMM system is lower, the amount of useful complementary information is higher, probably because of higher subspace dimensionality. Thus, the best size of the i-vectors used to describe the segments with examples of contextual phonemes seems to be around 30-60.

Note on MAP adaptation of means

We presented these experiments on meeting of speech@FIT group at BUT. We were told, that for such task (phoneme recognition), adapting only the means does not make sense, as this is commonly applied only to speaker models. This was at the end of our work, so we did not repeat the experiments. Note however, that both systems had such disadvantage, so this effect should be minor.

5.2 Application to speaker verification task

Motivated by the results obtained on phoneme verification task presented in previous section, we applied our feature extraction technique to speaker recognition. We assume, that if the features are useful in speech recognition, they should be also beneficial in this domain. Note, that it was also our first motivation – to come up with some general temporal feature extraction method for speaker recognition.

5.2.1 Database description

To test our hypotheses, we selected the NIST SRE 2008 evaluation data. Complete description of the data and evaluation conditions can be found in [34]. We have chosen only the male part, as all the state-of-the-art systems are using gender-dependent models. We could have used the female part, as this is more common due to the fact, that females are generally harder to recognize, but it was more convenient for us to choose the male part. In the core test of SRE 2008 evaluations (short2-short3) male part, there are 39433 trials, 1270 target speakers, 3798 evaluation utterances. All trials from the core test have to be evaluated by an evaluee, but the system performance is measured from the conditional subsets of the trials. All the conditions are described in [34], we will highlight here the following ones:

- **det1** – Interview Training and Test
- **det4** – Interview Training, Telephone Test
- **det6** – Telephone Training and Test
- **det7** – English Language, Telephone Training and Test
- **det0** – All trials (unofficial condition)

As development data, we used the male part from the following databases: NIST SRE 2004, 2005, 2006, Switchboard II Phase 2 and 3, Switchboard cellular part 1 and 2. In total, there were 15660 development utterances.

5.2.2 Experimental setup

Complete system diagram is depicted in Figure 5.2. The figure shows, that only the features are reparametrized, without any changes to the baseline system. Note also, that the reparametrization module is morphologically the same as the final i-vector system. The difference is, that reparametrization module works on segments rather than on utterances and that it uses only the subset of the whole development data. For practical reasons, we had to use only the 1 % (215 000 segments) of the whole development data to train the reparametrization module (total variability matrix).

Input signal parametrization

The parameters for the MFCC feature extraction are as follows (default LIA configuration):

- no preemphasis
- frame length 20 ms with 10 ms shift
- MFCC with 19 coefficients, log-energy, delta and double delta coefficients, frequency band 300 Hz - 3.4 kHz
- feature mask[2]: `0-18,20-50`

After MFCC extraction, voice activity detection (VAD) was performed, followed by utterance-level CMVN and feature warping.

Voice activity detection was performed using freely available phoneme recognizer from BUT[3]. Speech labels were created by merging all the segments belonging to phonemes.

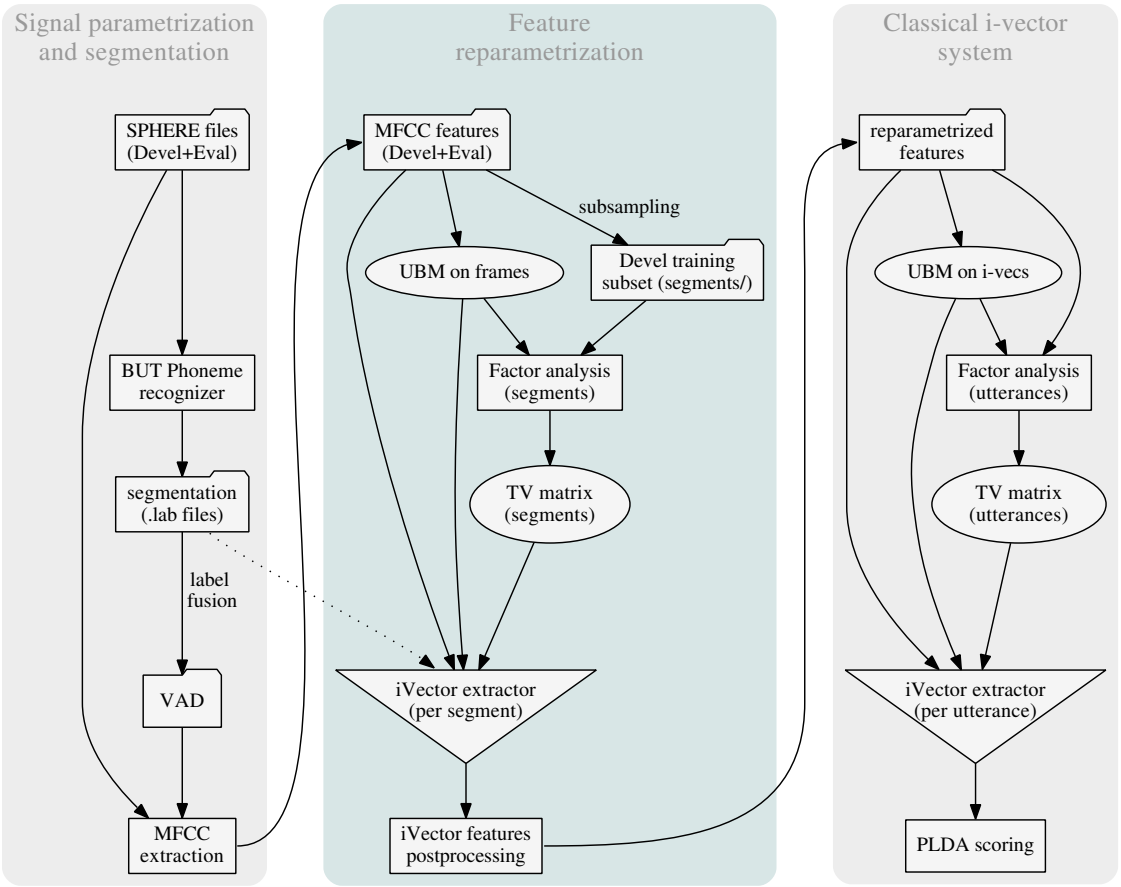


Figure 5.2: Flow-chart for the whole system.

[2]This mask defines which indices from feature vector should be used for further processing. Our mask selects all 19 MFCC coefficients, all delta coefficients, delta log-energy coefficient and first 11 acceleration coefficients.

[3]http://speech.fit.vutbr.cz/software/phoneme-recognizer-based-long-temporal-context

I-vector based feature reparametrization

Those are the main parameters used in this experiment:

- number of UBM components: 512 – this is common value of UBM size → supervector of size 25600.
- total variability matrix rank: 40 – this value seemed to be the best in the preceding experiments on phoneme verification task.
- number of *segments* used to train total variability matrix: 215k – this is around 1% of all development data. We could not use more data for practical reasons.

These parameters were fixed for all the experiments described later in this chapter.

Conditions for i-vector extraction

The main goal of this experiment was to find, how well our proposed system behaves in speaker recognition task. In addition, we wanted to see, whether the phoneme recognition backend is needed also for the evaluation data. This is the main criterion of all feature extraction techniques – if they depend on phoneme recognizer or not. This is depicted in Figure 5.2 by a dotted line. The two conditions are:

- **fixed** – no phoneme segmentation is used. The segments to be reparametrized are created from VAD segments by a fixed sliding window with parameters size and shift.
- **phnrec** – phoneme recognizer output is used. The segments to be reparametrized are the phonemes themselves.

These two conditions can be applied in two places. The first place is training the first i-vector extractor. As we describe in Section 4.2, by using different training data to estimate the total variability matrix, we can extract different features. In our experiments, we used only the *phnrec* condition to train total variability matrix. The second place is the feature reparametrization process itself, where the reparametrization of training and evaluation data is needed, based on some segmentation.

Last stage

In the last stage, the classical i-vectors are extracted from the reparametrized features and the session compensation technique using the two-covariance model. Main parameters:

- number of UBM components: 512 – we use in this place the same number of Gaussians as for the UBM trained on MFCC features. We assume, that distribution of reparametrized features will be again multimodal.
- size of i-vectors: 400 – we use this value as it has proven to be a good for i-vectors extracted from acoustic features.
- two-covariance model with length normalization

Reference baseline system

For a comparison, we used state-of-the-art i-vector system developed at LIA. In our experiments, we used exactly the same parameters and procedures for both baseline and the proposed system, except for the added feature reparametrization step in case of the new system. This means, that the reference baseline system should use also the phoneme recognizer based VAD. But as we will see in results, baseline system with energy based VAD (LIA baseline) performed better in general, so the **primary baseline system uses the energy based VAD** and it is the secondary baseline system that uses exactly the same portion of frames (phoneme recognizer based VAD) as uses the newly developed system.

5.2.3 Experimental results

In this section, we summarize the results of experiments performed using above mentioned systems – two baseline systems (baseline, baseline2) and our proposed system using i-vector feature reparametrization with two variants for segmentation speech (fixed, phnrec). The results from score level fusion are also presented to show, whether the reparametrized features are helpful or not.

Overall behavior for selected core conditions is depicted in Figure 5.3. We can see, that the systems using i-vectors as features are far behind the purely acoustic systems. In general, the i-vector reparametrization performs better for fixed condition than for phoneme aligned segmentation. All fused systems are drawn with dashed lines in Figure 5.3. We can see, that fusing our new system with baseline system improves the performance. Unfortunately, such gain can be obtained also by fusing the two versions of baseline system (black dashed line). The two baseline systems differ only in VAD – energy based or phoneme recognizer based. We try to explain this failure of our new system to capture temporal information useful for speaker discrimination in the next section.

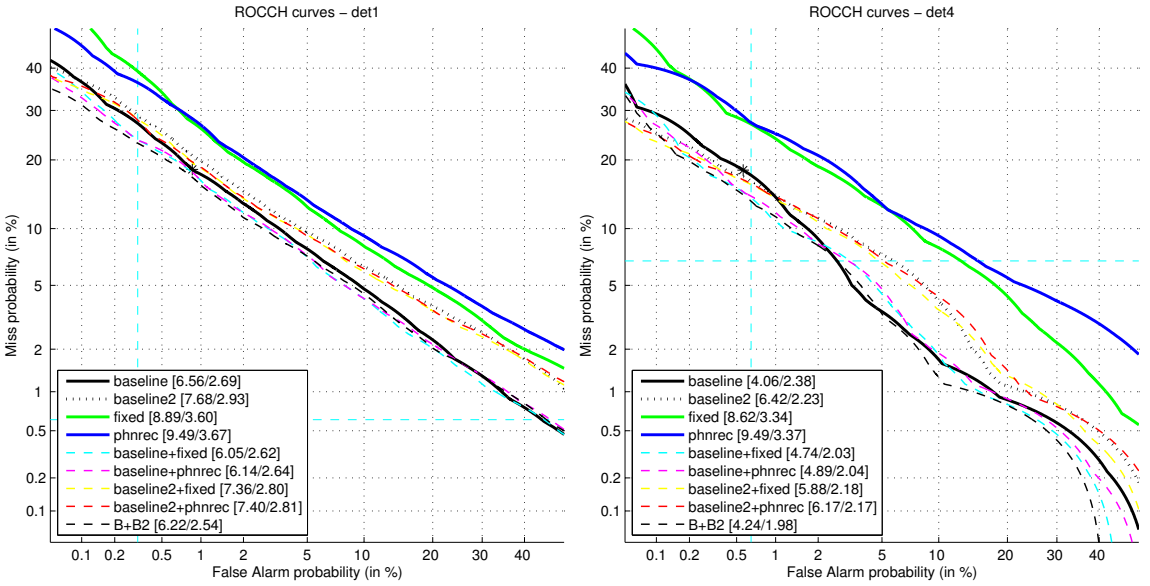
5.2.4 Discussion

The results obtained on speaker verification task are disappointing, and here we try to analyze, why the new system does not outperform the baseline, as expected. Our initial idea after having success with preliminary experiments on phoneme verification was, that if there is a gain for phoneme verification, we can expect also a gain for speaker recognition task. This was probably the main mistake.

Normally, there are mainly two variabilities in the space of i-vectors – session variability and speaker variability. For the i-vectors extracted from short segments (like we did in our experiments), there is in addition also the phonetic variability and it is clear that this is the largest one. If we use the i-vector paradigm to compress the segment information into 40 dimensional vector, maybe the useful speaker discriminative temporal information is completely outvoiced by the acoustic and phonetic variability.

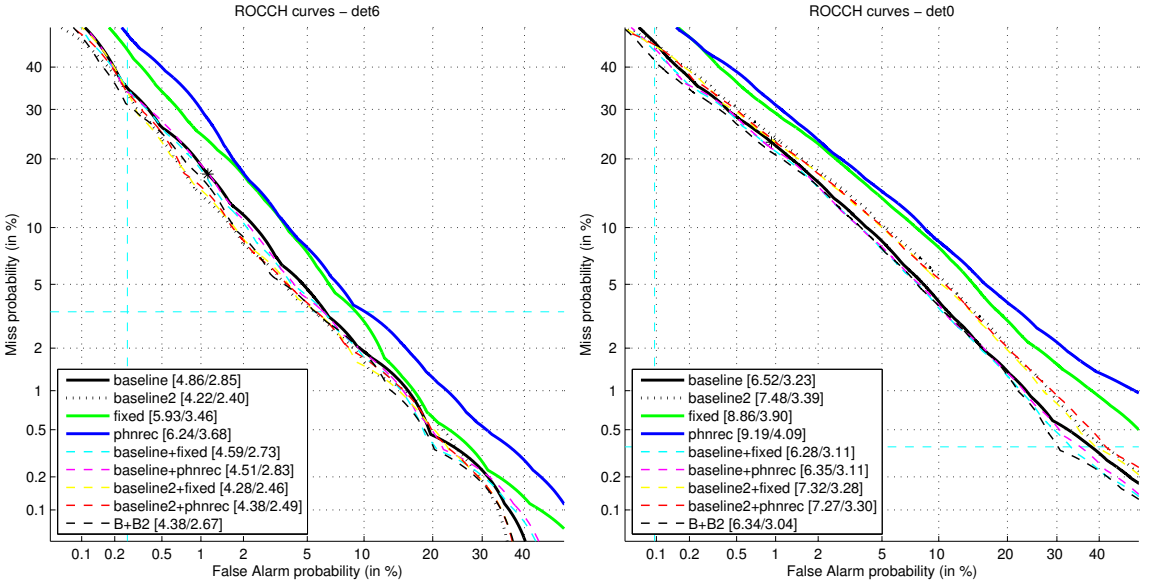
There are studies [33] which show the proportions of different variabilities in spectral domain. The biggest portion of variance belongs to phoneme and context variability. For the channel and speaker variability there is 16% and 10 %, respectively. In our work, we wanted to make use of temporal speaker variability, that is probably too small to be even visible for our classifier.

We show the results of the reparametrization process for one NIST utterance in Figure 5.4 to present the nature of extracted features. We can see that the features are highly uncorrelated. Note however, that every frame in this figure corresponds to one segment



(a) Condition 1 – interview training and test

(b) Condition 4 – interview training and telephone test



(c) Condition 6 – telephone speech

(d) Condition 0 – all trials from core test

Figure 5.3: ROCCH curves for single and fused systems on different core test conditions ($\%EER/C_{Det}$). Dashed vertical and horizontal lines define the area, where there are at least 30 classification errors for the baseline system (Doddington's rule of 30).

(i.e. approximately to one phoneme) and so this can be expected, as the phoneme class changes radically for each frame.

From this picture it is also clear, that the feature distribution has changed a lot. We expected this behaviour, as there is an assumption of gaussianity in total variability model – the i-vectors are assumed to be normally distributed among $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. We thought, that using segments of defined content for total variability matrix training, we can achieve that after extraction, the distribution of the i-vector based features will be again multimodal. This was probably not fulfilled and so it explains also the problems we had with training the UBM model on such features.

We spent a lot of time examining, why the single systems based on reparametrized features are not performing at least as good as baseline. We conclude, that this is caused by a combination of previous imperfections.

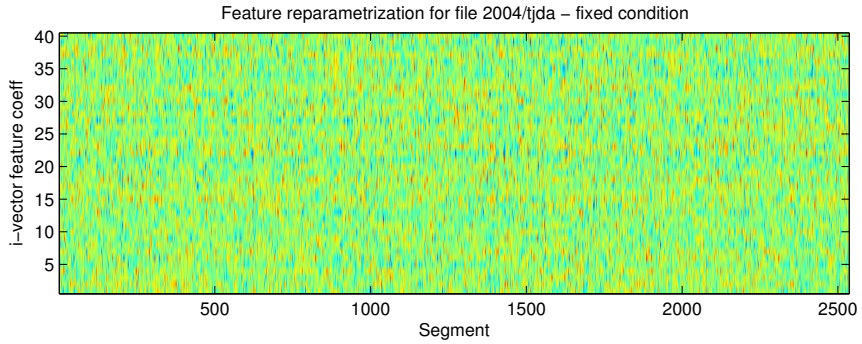


Figure 5.4: Visualization of the reparametrized features for NIST 2004/tjda utterance. The image is similar also for the phnrec condition.

Computational complexity

From the practical point of view, proposed feature reparametrization process requires very high demands for computation power. We realized this limitation in the beginning of our work. Please note, that this is a research project, so speed was not the main evaluation parameter.

Firstly, our system has to run phoneme recognizer or some pseudo-phoneme segmentation. Thanks to this, our approach places itself into the category of „slow“ feature extraction methods that requires segmentation.

During the development phase, the UBM and total variability matrix has to be trained. This is comparable to standard i-vector system, but the number of training sessions (and thus the memory complexity of this step) is much bigger, as the segments are considered as utterances.

The reparametrization process itself takes also a good amount of time, even if the frame rate is lowered (wrt. acoustic features). One iteration of i-vector estimation procedure has to be run for each segment.

Chapter 6

Conclusions and future work

In this work we proposed new approach to temporal feature extraction using the i-vector paradigm. Preliminary experiments were presented to show, that such parametrization is possible and could be beneficial for phoneme verification task. By using our approach on this task, in comparison with the baseline acoustic frame level system, significant improvement was obtained. This also corresponds with the success of similar method [6], where the i-vectors were used to represent HMM-states.

On the other hand, our main objective – to develop temporal feature extraction method suitable for speaker verification – has not been really resolved. The experiments on NIST SRE 2008 database have shown, that such features are not suitable for further speaker modeling, as they convey mostly the phonetic variability, which is not very useful for speaker discrimination. Even more, we think, that such features are even worse, as their distribution is no longer multimodal and without feature clusters corresponding to broad acoustic classes. Such clusters are essential to be able to use speaker models based on GMM.

Future work

As there are always new possibilities and ideas, we would like to give some directions of future work on this subject. Firstly, it could be interesting to try lower the unneeded variability from segmental i-vectors somehow. We think that phoneme dependent extraction or Joint Factor Analysis could be used for this purpose. Secondly, the chance should be given to the results obtained on phoneme verification task. Maybe our feature reparametrization can not be used for directly for speaker recognition, but is more interesting for speech recognition, as a context-capturing mechanism.

We realize, that we do not give any comparisons with conventional temporal feature extraction methods. This could be easy and straightforward, to compare the i-vector based representation of short segments with other methods, where the analysis window ranges over multiple speech frames, like in the case of bottleneck features.

Bibliography

[1] Md Jahangir Alam, Pierre Ouellet, Patrick Kenny, and Douglas O'Shaughnessy. Comparative evaluation of feature normalization techniques for speaker verification. In *Proceedings of the 5th International Conference on Advances in Nonlinear Speech Processing*, NOLISP'11, pages 246–253, Berlin, Heidelberg, 2011. Springer-Verlag.

[2] E. Ambikairajah, Haizhou Li, Liang Wang, Bo Yin, and V. Sethu. Language identification: A tutorial. *Circuits and Systems Magazine, IEEE*, 11(2):82–108, 2011.

[3] Homayoon Beigi. *Fundamentals of speaker recognition.* New York, NY: Springer. lxi, 942 p., 2011.

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

[5] Tobias Bocklet, Andreas Maier, and Elmar Nöth. Text-independent speaker identification using temporal patterns. In *Proc. of Text, Speech and Dialogue, 10th International Conference, TSD 2007, Pilsen, Czech Republic, September 3-7, 2007*, volume 4629 of *Lecture Notes in Computer Science*, pages 318–325. Springer, 2007.

[6] M. Bouallegue, E. Ferreira, D. Matrouf, G. Linares, M. Goudi, and P. Nocera. Acoustic modeling for under-resourced languages based on vectorial HMM-states representation using subspace gaussian mixture models. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 330–335, Dec 2012.

[7] Pierre-Michel Bousquet, Anthony Larcher, Driss Matrouf, Jean-Francois Bonastre, and Oldřich Plchot. Variance-spectra based normalization for i-vector standard and probabilistic linear discriminant analysis. In *Proceedings of Odyssey 2012: The Speaker and Language Recognition Workshop*, pages 157–164. ISCA, 2012.

[8] Pierre-Michel Bousquet, Driss Matrouf, and Jean-François Bonastre. Intersession compensation and scoring methods in the i-vectors space for speaker recognition. In *Interspeech*, pages 485–488. ISCA, 2011.

[9] Niko Brümmer. *Measuring, refining and calibrating speaker and language information extracted from speech.* PhD thesis, Department of Electrical and Electronic Engineering, University of Stellenbosch, 2010.

[10] Niko Brümmer and Edward de Villiers. The speaker partitioning problem. In *Proceedings of Odyssey 2010: The Speaker and Language Recognition Workshop, Brno, Czech Republic, June 28 - July 1, 2010*, page 34. ISCA, 2010.

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

[12] Niko Brümmer, Lukáš Burget, Jan Černocký, Ondřej Glembek, František Grézl, Martin Karafiát, David Leeuwen van, Pavel Matějka, Petr Schwarz, and Albert Strasheim. Fusion of heterogeneous speaker recognition systems in the stbu submission for the NIST speaker recognition evaluation 2006. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2072–2084, 2007.

[13] Lukáš Burget, Oldřich Plchot, Sandro Cumani, Ondřej Glembek, Pavel Matějka, and Niko Brümmer. Discriminatively trained probabilistic linear discriminant analysis for speaker verification. In *Proc. of the 2011 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011*, pages 4832–4835. IEEE Signal Processing Society, 2011.

[14] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, and P. A. Torres-carrasquillo. Support vector machines for speaker and language recognition. *Computer Speech and Language*, 20:210–229, 2006.

[15] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011.

[16] Najim Dehak, Pedro A. Torres-Carrasquillo, Douglas Reynolds, and Reda Dehak. Language Recognition via I-Vectors and Dimensionality Reduction. In *Interspeech 2011*, pages 857–860, Florence, Italy, August 2011.

[17] Sadaoki Furui. Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(2):254–272, 1981.

[18] Ondřej Glembek. *Optimization of Gaussian Mixture Subspace Models and related scoring algorithms in speaker verification.* PhD thesis, Brno University of Technology, Faculty of Information Technology, 2012.

[19] F. Grezl, M. Karafiat, S. Kontar, and J. Cernocky. Probabilistic and bottle-neck features for LVCSR of meetings. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–757–IV–760, April 2007.

[20] Hynek Hermansky and Sangita Sharma. TRAPS – classifiers of temporal patterns. In *Proceedings of* 5^{th} *International Conference On Spoken Language Processing, ICSLP 98*, pages 1003–1006, 1998.

[21] Qin Jin and Thomas Fang Zheng. Overview of front-end features for robust speaker recognition. In *Proceedings of APSIPA ASC 2011 Xi'an*, 2011.

[22] Sachin Kajarekar, Luciana Ferrer, Kemal Sönmez, Jing Zheng, and Elizabeth Shriberg. Modeling NERFs for speaker recognition. In *Proc. Odyssey 2004: The Speaker and Language Recognition Workshop*, 2004.

[23] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel. Joint factor analysis versus eigenchannels in speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4):1435–1447, 2007.

[24] Patrick Kenny. Bayesian speaker verification with heavy-tailed priors. In *Proceedings of Odyssey 2010: The Speaker and Language Recognition Workshop, Brno, Czech Republic, June 28 - July 1, 2010*, page 14. ISCA, 2010.

[25] T. Kinnunen. Joint acoustic-modulation frequency for speaker recognition. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I, 2006.

[26] Tomi Kinnunen and Haizhou Li. An overview of text-independent speaker recognition: From features to supervectors. *Speech Commun.*, 52(1):12–40, January 2010.

[27] Tomi Kinnunen, Chin Wei, Eugene Koh, Lei Wang, Haizhou Li, and Eng Siong Chng. Temporal discrete cosine transform: Towards longer term temporal features for speaker verification.

[28] Marcel Kockmann. *Subspace modeling of prosodic features for speaker verification.* PhD thesis, Brno University of Technology, Faculty of Information Technology, 2011.

[29] I. Magrin-Chagnolleau, G. Durou, and F. Bimbot. Application of time-frequency principal component analysis to text-independent speaker identification. *Speech and Audio Processing, IEEE Transactions on*, 10(6):371–378, 2002.

[30] David González Martínez, Oldřich Plchot, Lukáš Burget, Ondřej Glembek, and Pavel Matějka. Language recognition in ivectors space. In *Proceedings of Interspeech 2011*, volume 2011, pages 861–864. ISCA, 2011.

[31] Driss Matrouf, Nicolas Scheffer, Benoit G. B. Fauve, and Jean-François Bonastre. A straightforward and efficient implementation of the factor analysis model for speaker verification. In *Interspeech*, pages 1242–1245. ISCA, 2007.

[32] Pavel Matějka, Petr Schwarz, Hynek Hermansky, and Jan Černocký. Phoneme recognition using temporal patterns. In *Proc. of Text, Speech and Dialogue, 10th International Conference, TSD 2007, Pilsen, Czech Republic, September 3-7, 2007*, volume 2807 of *Lecture Notes in Computer Science*, pages 198–205. Springer Berlin Heidelberg, 2003.

[33] Sachin Kajarekar Narendranath, Sachin Kajarekar, Narendranath Malayath, and Hynek Hermansky. Analysis of speaker and channel variability in speech, 1999.

[34] National Institute of Standards and Technology. *The NIST Year 2008 Speaker Recognition Evaluation Plan.* NIST, 2008.

[35] Jason Pelecanos and Sridha Sridharan. Feature warping for robust speaker verification. In *Proceedings of Odyssey 2001: The Speaker and Language Recognition Workshop.* ISCA, 2001.

[36] Simon J. D. Prince. Probabilistic linear discriminant analysis for inferences about identity. In *ICCV*, 2007.

[37] Douglas A. Reynolds. Speaker identification and verification using gaussian mixture speaker models. *Speech Communication*, 17(1-2):91–108, 1995.

[38] Douglas A. Reynolds, George R. Doddington, Mark A. Przybocki, and Alvin F. Martin. The NIST speaker recognition evaluation - overview methodology, systems, results, perspective. *Speech Communication*, 31(2-3):225–254, June 2000.

[39] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. Speaker verification using adapted gaussian mixture models. In *Digital Signal Processing*, 2000.

[40] Pedro A. Torres-Carrasquillo, Elliot Singer, Mary A. Kohler, Richard J. Greene, Douglas A. Reynolds, and John R. Deller Jr. Approaches to language identification using gaussian mixture models and shifted delta cepstral features. In John H. L. Hansen and Bryan L. Pellom, editors, *Interspeech*. ISCA, 2002.

[41] Bing Xiang, Upendra V. Chaudhari, Jiri Navratil, Ganesh N. Ramaswamy, and Ramesh A. Gopinath. Short-time gaussianization for robust speaker verification. In *Proc. ICASSP*, pages 681–684, 2002.

[42] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. *The HTK Book, version 3.4*. Cambridge University Engineering Department, Cambridge, UK, 2006.

Appendix A

Implementation details

A.1 Libraries and toolkits used

We used the following toolkits to run and evaluate our experiments:

- **ALIZE** – toolkit for biometric authentication[1]
- **Bosaris Toolkit** – MATLAB code for calibrating, fusing and evaluating scores from binary classifiers[2]
- **SPro** – signal processing toolkit, feature extraction[3]

We also used the MATLAB® for visualization purposes.

A.2 DVD contents

Here we highlight the important content of the DVD attached to this thesis:

`experiments/`	main directory containing source code and scripts
`ester/`	scripts and some data (indexes, models) used in experiments with phoneme verification on ESTER database. Three main directories are `data/`, `baseline/` and `iGMM_suite`.
`sre08/`	scripts used in experiments with speaker verification on SRE 2008.
`TV_*/`	total variability matrix training (fixed, phnrec)
`reparam_*/`	feature reparametrization (fixed, phnrec)
`FA_reparam_fixed/`	second stage i-vector extraction and PLDA scoring (fixed). Analogous directory for phnrec condition can be found in `reparam_phnrec/` folder.
`tools/`	general purpose scripts (UBM training, reparametrization)
`iVector_intro.MOV`	short video created as an introduction to iVectors
`poster/`	poster source files
`tex/`	LaTeX sources for this thesis

[1] http://alize.univ-avignon.fr/index_en.html
[2] https://sites.google.com/site/bosaristoolkit/
[3] http://www.irisa.fr/metiss/guig/spro/

A.3 How to use it?

The main objective for us were the results, and how to get them quickly, so we apologize for dirty and messy code, with only a little documentation. There is nothing like graphical front-end. Everything is done by calling scripts from command line.

If there is someone interested in this feature extraction technique, we advise him to start coding from scratch, as this will be definitely faster than analyzing and adapting our scripts. We have not created any form of ready to use toolkit.

We use heavily the ALIZE toolkit. For the needs of i-vector reparametrization, we have created two additional tools that can be used directly:

- `iXallseg` – feature reparametrization based on segments from external label file.
- `iXwindow` – feature reparametrization based on segments selected using sliding window mechanism.

Except ordinary command line parameters[4], these tools expect filename of saved total variability matrix. These tools can be found in `experiments/tools/LIA_SpkDet/` folder. The process of reparametrization together with short description of related scripts can be found in file `experiments/tools/reparametrization/README.txt`.

[4] http://mistral.univ-avignon.fr/mediawiki/index.php/Config_file_guide