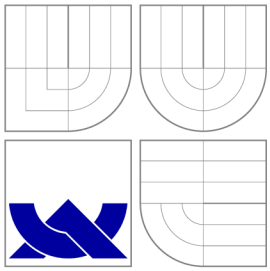
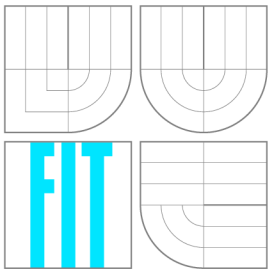


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

GRAFICKÉ A VIDEO PŘÍZNAKY V ROZPOZNÁVÁNÍ MLUVČÍHO

COMPUTER GRAPHICS AND VIDEO FEATURES FOR SPEAKER RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

RADEK FÉR

VEDOUCÍ PRÁCE
SUPERVISOR

JAN ČERNOCKÝ, doc. Dr. Ing.

BRNO 2012

Abstrakt

Tato práce popisuje netradiční metodu rozpoznávání řečníka pomocí příznaků a alogoritmů používaných převážně v počítačovém vidění. V úvodu jsou shrnuty potřebné teoretické znalosti z oblasti počítačového rozpoznávání. Jako aplikace grafických příznaků v rozpoznávání řečníka jsou detailněji popsány již známé BBF příznaky. Tyto jsou vyhodnoceny nad standardními řečovými databázemi TIMIT a NIST SRE 2010. Experimentální výsledky jsou shrnuty a porovnány se standardními metodami. V závěru jsou jsou navrženy možné směry budoucí práce.

Abstract

We describe a non-traditional method for speaker recognition that uses features and algorithms used mainly for computer vision. Important theoretical knowledge of computer recognition is summarized first. The Boosted Binary Features are described and explored as an already proposed method, that has roots in computer vision. This method is evaluated on standard speaker recognition databases TIMIT and NIST SRE 2010. Experimental results are given and compared to standard methods. Possible directions for future work are proposed at the end.

Klíčová slova

rozpoznávání mluvčího, Boosted Binary Features (BBF), boosting, lokální řečové příznaky

Keywords

speaker recognition, Boosted Binary Features (BBF), boosting, localized speech features

Citace

Radek Fér: Computer Graphics and Video Features for Speaker Recognition, bakalářská práce, Brno, FIT VUT v Brně, 2012

Computer Graphics and Video Features for Speaker Recognition

Prohlášení

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

.....
Radek Fér
May 16, 2012

Poděkování

Děkuji panu doc. Dr. Ing. Janu Černockému za počáteční nasměrování na tuto zajímavou oblast. Rád bych mu poděkoval za vedení práce, všemožnou podporu a jeho vstřícnost. Dále děkuji panu Ing. Oldřichu Plchotovi za pomoc a konzultace týkající se současných technik rozpoznávání řečníka a řečových databází. Děkuji tímto také mým rodičům za lásku a podporu ve studiu.

© Radek Fér, 2012.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Contents

Page

1 Introduction **2**
- 1.1 Related work . . . 2
- 1.2 Organization of this work . . . 2

2 Theoretical background **3**
- 2.1 Speaker recognition . . . 3
- 2.2 Features for speaker recognition . . . 4
- 2.3 Computer graphics features . . . 4
- 2.4 Boosting . . . 6
- 2.5 Evaluation metrics and data . . . 6
- 2.6 Score normalization . . . 8

3 Overview of Boosted Binary Features **9**
- 3.1 Relation of BBF to computer graphics features . . . 9
- 3.2 Feature description and speaker modeling . . . 9
- 3.3 Open questions . . . 12
- 3.4 Complete system . . . 14

4 Experimental setup and results **15**
- 4.1 Reference systems . . . 15
- 4.2 TIMIT . . . 16
- 4.3 NIST SRE 2010 . . . 22
- 4.4 Results and discussion . . . 26

5 Conclusions and future work **27**
- 5.1 Future work . . . 27

Bibliography **28**

A Implementation details **31**
- A.1 Directory structure . . . 31
- A.2 How to use it? . . . 31
- A.3 Libraries and toolkits used . . . 32

Chapter 1

Introduction

The need for accurate speaker recognition is increasing today. It is not just an academic problem for a long time. Telephone companies, banks and others (including police and intelligence agencies of course) would like to use speaker recognition systems (SRS) for different purposes: voice verified authentication, call tracking or automatic labeling of voice data, etc. Today's state-of-the-art SRSs use mostly short-term spectral features, such as MFCC coefficients, for speaker modeling. We investigate usage of features known from computer vision domain in the task of speaker recognition in this work. We focus on the Boosted Binary Features (BBF) as already existing features usable for this task.

1.1 Related work

Probably the most recent attempt to exploit computer graphics-like features in speech domain is work around Boosted Binary Features (BBF), done mainly by Anindya Roy [23][24][25][26]. In this work, we are following what has been already done, using these sources.

There are also other studies investigating different approaches of getting speaker features from signal spectrum [18], but mostly, these features are global/holistic (i.e. not localized or part-based) – they are not similar to features known from computer vision domain (these are almost always localized). Because of this, these features are not considered in our work.

1.2 Organization of this work

This paper is organized as follows: Needed theoretical background is given in Chapter 2. We also present an overview of features for speaker recognition, together with features used in computer vision in that chapter. The BBF approach is described in Chapter 3. At the end of the chapter, we introduce several questions. These questions are answered in Chapter 4, where we summarize the results obtained during our work. You will find the results of the BBF system evaluations on the TIMIT and the NIST SRE 2010 databases there. The results are discussed afterwards. We conclude and give directions for future work in Chapter 5. There is also Appendix A, where practical information about our BBF system implementation can be found.

Chapter 2

Theoretical background

The theory employed in our work is presented in this chapter. Well-explored things of our interest are summarized here. We start with basic concepts of speaker recognition systems.

2.1 Speaker recognition

Speaker recognition tries to answer the question: „Who is speaking?“ There are 2 different tasks: *speaker identification* and *speaker verification/detection*. We illustrate the difference in Fig. 2.1.

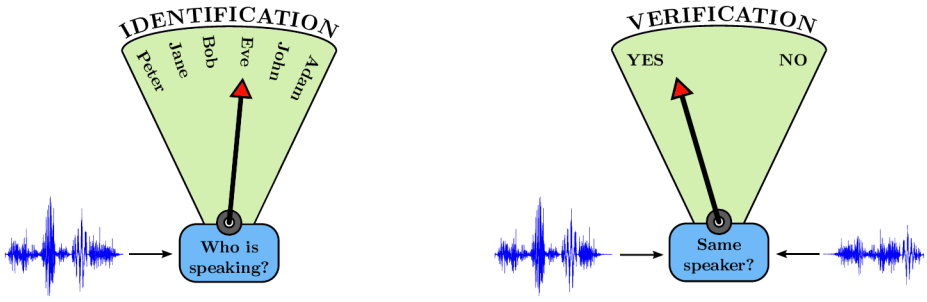


Figure 2.1: Automatic speaker identification (ASI) and verification (ASV).

Although the tasks are different, both makes use of the same kind of information – speaker identity. The corresponding automatic speaker recognition systems (either for verification or identification) must be able to express such a quality in some way. We describe how to do it in the next section.

We can divide speaker recognition also into *text dependent* and *text independent*. Text independent recognition is much more challenging [15] and can be used in scenarios without client cooperation. For example speaker detection on a telephone line, speaker diarization, caller identity checking etc.

Today, we are still looking for the best technique, how to extract information from a speech signal, that can be used as an input to a pattern recognition algorithms. We introduce this topic in the next section.

2.2 Features for speaker recognition

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. But also, speech signal carries everything else together with speaker information, like speaker mood, channel information, linguistic content, noise and so on. For the speaker recognition task, we want to use only relevant part of the input signal.

We must firstly transform the speech signal into so-called *feature space*, where computers (classification algorithms) can see the actual features. This is called **feature extraction**. Extracted features can be divided into low level features (pitch, timbre) and higher level speaker features like intonation (so-called prosodic features) and even higher features like idiolect (selection of speaker specific words or phrases) [15]. We will consider only low level features in the following text, not going into the higher ones.

All the state-of-the-art SRSs are using to some extent these low level features, because they are easy to extract and model. They usually benefit from the knowledge of human auditory and vocal tract system.

We give a short description of currently the most often used low level features for speaker recognition – the MFCC coefficients – in this section. Nevertheless, there are also other (namely Linear Prediction Cepstral Coefficients – LPCC features), we refer to a nice and complete overview in [15].

- **MFCC** – Mel-Frequency Cepstral Coefficients are famous speech features used very often for automatic speech/speaker recognition. They can be described as a short-term cepstral features that use Mel-filter banks. After the signal is processed by the STFT (Short Term Fourier Transform), Mel-banks transforms linear frequency scale into a scale similar to the scale of human ear perception capabilities. Logarithm of this Mel-spectrum is taken and Discrete Cosine Transform (DCT) is applied. Final MFCC features are obtained by retaining about 12-15 lowest DCT coefficients [15].

  It is very often and useful, after feature extraction, to add Δ and $\Delta\Delta$ (or even higher-order delta) coefficients to MFCC. By using this technique, features become temporal – they carry also time context. One immediate disadvantage of the global nature of MFCC is that it reduces noise-robustness: addition of noise in a small sub band affects the entire representation [4]. This can be partly suppressed by cepstral mean subtraction (CMS) [3] (which removes slowly varying convolutive noise) and other advanced techniques.

2.3 Computer graphics features

There are quite different tasks in computer vision in contrast to speaker recognition, but there are analogies. The most analogous to speaker recognition is face recognition, where we want to find out the identity of a person from an image. We can also compare phoneme detection or keyword spotting to computer vision object detection. These ideas (to treat phonemes, words and such as *sound objects* in time-frequency plane) can be found in [1][17].

This section describes some of commonly used graphics features and their characteristics. What follows is mainly an excerpt from a similar overview in [23]. We try to highlight characteristics, that could be interesting for speaker recognition task.

- **Haar features** – this type of features is extensively used in image object detection. It is typically used in Viola & Jones object detection framework [27]. Features are defined as a difference of sums of two square regions in the image. This difference is then compared to a learned threshold and serves as a weak prediction of presence of some object. An example of such regions (features) is in Fig. 2.2. These regions are then scaled to different sizes and can be also tilted ($\pm 45°$). In the training phase, *boosting* algorithms are used to select the best features (i.e. the position in the image, scale, tilt and a threshold for each feature).

Figure 2.2: Common Haar feature regions

- **Local Binary Patterns (LBP)** – these features are known for its invariance to illumination. Feature extraction is shown in Fig. 2.3. In principle, the neighborhood is thresholded by the value of the center pixel and the binary values are then multiplied by the weights given for the corresponding pixel and summed. Resulting LBP code is not used as a single feature, but a histogram of these codes is computed over an image region and then used as a feature. There are many variations of the basic 3×3 neighborhood. It was proposed in [20].

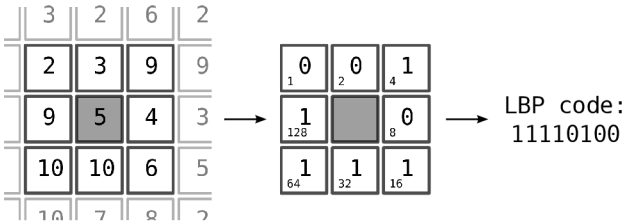


Figure 2.3: The way, how the LBP features are extracted

- **Fern features** – these features were proposed in [21] and could be seen as a generalization of the LBP features. Instead of considering regular neighborhood, these features are computed by comparing the „center“ pixel with an arbitrary other pixel in the image. This leads to much larger feature set.

- **Descriptor-based features** – used in computer vision to detect and describe *points of interest* in the image. They are used for camera calibration, 3D reconstruction, object detection and other tasks [2]. As a representative, we can mention SURF (Speeded-Up Robust Features) and SIFT (Scale Invariant Feature Transform) descriptors. Computation is quick but not easy to describe (the exaction has usually multiple stages like detection, indexing etc.), so we refer to [2].

  Intuitively, these features, modified for use in speech domain, could be suitable rather for speech recognition (phoneme detection) than for speaker recognition. We mention them here because of this analogy example.

2.4 Boosting

In the following chapters, we will be dealing with boosting. This section provides some basic information about boosting. For more in depth description, please, see [11][12].

Boosting is a way of combining the performance of many *weak classifiers* to produce a powerful *committee* or *strong classifier* [12] in an iterative way. The only requirement for a weak classifier is, that it should be „better than random guessing“.

The most famous boosting algorithm is the aDAbOOST. For example, it is being used for feature selection in Viola&Jones object detection framework [27], where Haar features are utilized. The term AdaBoost is a shorthand for Adaptive Boosting.

In this work, we use the variant of AdaBoost called *Discrete AdaBoost with weighted re-sampling* [23]. Full algorithm with explanation is in Section 3.2.2. There are many other variants of classical AdaBoost – Real AdaBoost, Gentle AdaBoost, Total Corrective AdaBoost ...

There are studies [7][9] showing bad behaviour of the AdaBoost on data with label noise and outliers, because boosting tends to assign the examples to which noise was added much higher weight than other examples. As a result, hypotheses generated in later iterations cause the combined hypothesis to over-fit the noise [9]. In other words, the AdaBoost assumes, that the data are linearly separable.

There are boosting algorithms without this problem, namely the BrownBoost [9] and its most recent variant RobustBoost. These algorithms do not use convex optimization (that is responsible for mentioned bad behavior of the AdaBoost [10]), but use other optimization techniques to minimize misclassification loss.

2.5 Evaluation metrics and data

To select the best system, we must be able to objectively compare the two. To be objective, some common measure of system accuracy must exist. Also, the need for common data set is important. Results obtained on non-standard or proprietary corpus are obscure and not comparable with other ones.

This section describes, how speaker recognition systems can be evaluated, and describes standard speech corpora we are using in this work.

2.5.1 Evaluation metrics

We describe standard ways, how to evaluate speaker recognition systems, in this section.

The output of a recognizer is a score. It is a numerical measure (estimation) of how confident the recognition system is, given some trial. A trial is a pair of a test utterance and a claimed identity. The score is referred as a *soft decision* sometimes. When we need a *hard decision*, we must set some threshold for scores to obtain final class labels. We describe here, how we can interpret the scores and hard decisions.

Things are little easier for speaker *identification* task. There is only one type of system error – misclassification. For such purpose, we can use *probability of misclassification*, that is simply a number of incorrectly classified trials divided by a total number of trials.

For speaker *verification*, we can not use such simple metric. There are two cases, where the system can err [16]. See table 2.1 for all possible states.

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

Table 2.1: Possible outcomes for speaker verification

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

$$P_{miss} = \frac{\text{number of FR}}{\text{number of client trials}}$$

$$P_{fa} = \frac{\text{number of FA}}{\text{number of impostor trials}}$$

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.

P_{fa} and P_{miss} are not usually 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.

Overall system behavior can be visualized by the *ROC* (Receiver Operating Characteristic) or better with the *DET* (Detection Error Trade-off) curve [16].

For quick comparison of systems, we can pick some well defined point on the DET curve. This is typically the *EER (Equal Error Rate)*. The point, where $P_{miss} = P_{fa}$.

2.5.2 Standard speaker databases

TIMIT

This is an old standard speech database sponsored by DARPA. It contains a total of 6300 sentences (2-3 s in length), 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States. Each sentence is transcribed and phonetic content is given with timing. TIMIT is an ideal input for any SRS, because there is no inter-session variability, sampling frequency is sufficiently high (16 kHz) and the sentences are phonetically rich. Also, high quality microphones were used for recordings.

This database is subdivided into suggested test and train sets. This division is intended for speech recognition purposes. To use this database for speaker verification evaluations, we can use division from [22] (this division is also described in Section 4.2.2).

NIST SRE

National Institute of Standards and Technology Speaker Recognition Evaluation[1] is held every year and provides an opportunity for researchers to compare their systems. Data corpus is available for participants only, and is reasonably big. Every year, there are some new evaluation conditions and more difficult data. Unlike TIMIT, this data can be considered as real (telephone quality, different channel effects, conversational excerpts) and more challenging. Results of recognition system on such database are much more representative.

[1] http://www.itl.nist.gov/iad/mig/tests/sre/

Also, there are results available for recent evaluations, involving performance of today's state-of-the-art SRSs, that one can use for up-to-date comparison.

In this work, we use subset of data from SRE 2010 and SRE 2008. In each database, there are several training and testing conditions. One combination of training and testing is called *core* condition and is mandatory for every participant. This data can be further subdivided into *all trials involving only telephone speech in training and test*, *all trials involving interview speech from the same microphone type in training and testing* and so on. We are using for our purposes the first mentioned condition, also referred as *tel-tel* or *condition 5*.

Database	**Male/female speakers**	**Training**	**Testing**	**Target/nontarget trials**	F_s
TIMIT	112/56	20 s	2-3 s	336/280560	16 kHz
SRE 2010 core condition 5	1906/2361	2.5 min	2.5 min	3704/233077	8 kHz

Table 2.2: Basic parameters for TIMIT and NIST SRE 2010 condition 5 evaluation data. The TIMIT database was subdivided into train and test as in [22]. Durations for training and testing are only approximate.

2.6 Score normalization

In some cases it is useful to perform score normalization. The problem here is caused by different dynamic ranges of classifier responses to input data.

There are two basic types of score normalization [15]. The Z-normalization (zero normalization) is used to align scores between different speakers. This can be done in the enrollment phase (offline). It is complementary to setting client specific detection thresholds. The T-normalization (test normalization) aligns scores (classifier responses) between different types of test segments (f.e. segments with different levels of noise). By normalization, we mean

$$s' = \frac{s - \mu}{\sigma}$$

In this work we are using only Z-norm, so here is the detailed description. For Z-normalization, the μ (mean) and σ (standard deviation) are estimated for every client model from a set of impostor utterances, called cohort set. This set is the same for every client.

Chapter 3

Overview of Boosted Binary Features

This chapter describes recent approach for feature extraction and speaker modeling called *Boosted Binary Features (BBF)*. It was proposed in several documents by Anindya Roy [23][24][25][26]. It is novel, because it is recent, but also because the unusual way of speech feature extraction it uses. After short introduction to computer graphics features in previous chapter, we start this chapter by looking at how it is similar to features known from computer vision domain, and then we describe in depth, how such features can be extracted from speech signal.

3.1 Relation of BBF to computer graphics features

As it can be deduced from [23] (chapter 3), the idea behind BBF comes from the idea of Fern features, and these can be thought as a generalization of the Local Binary Patterns (see Section 2.3), both used in computer vision domain. They all (LBP, Fern, BBF) are *localized*, which means, that a feature is dependent only on some small region (subsection) of an input signal. This property has several nice consequences, like noise robustness (both in speech and vision).

On the other hand, almost all known features used for speech description today are *holistic* – single feature is influenced by the entire input signal (like MFCC and LPCC).

For clarification, we can write down (very roughly) something like this:

$$\text{LBP} \xrightarrow{\text{generalization}} \text{Fern} \xrightarrow{\text{vision}\rightarrow\text{speech}} \text{BBF}$$

3.2 Feature description and speaker modeling

This section describes, how BBF features looks like and how we use them. It may be a bit confusing at the beginning, what we call a *feature* here. Typically, in speaker recognition, we talk about features, after we have transformed the input data into some higher dimension space and then we use these features, to fit some probabilistic model or for discriminative training. In the BBF approach, „feature“ (as in Boosted Binary *Features*) has a wider sense. This is because feature extraction and speaker modeling are linked. Let's look at how.

3.2.1 Definition of BBF

One single feature is defined as a difference of 2 points in spectro-temporal plane and this difference is thresholded by some threshold. Hence they are binary. The spectro-temporal plane is created by transforming input speech signal into spectrogram. To keep number of possible features sane, we consider only points inside a sliding window, that we will refer later as a spectro-temporal matrix.

This matrix has dimensions N_F (N_F equals to a number of unique frequency points in the symmetric magnitude spectra obtained by the Fourier transform) and N_T (number of stacked spectral vectors).

In this text, we will follow [23] and let the spectro-temporal matrix to be only single spectral vector ($N_T = 1$), thus completely omitting temporal context in task of speaker recognition. This should not be critical, even that temporal context is known to be advantageous to MFCC based systems in form of Δ-coefficients, for speaker recognition [15].

The feature is defined:

$$f_i(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x}(k_{i,1}) - \mathbf{x}(k_{i,2}) \geq \theta_i \\ 0 & \text{if } \mathbf{x}(k_{i,1}) - \mathbf{x}(k_{i,2}) < \theta_i \end{cases}, \tag{3.1}$$

where $\mathbf{x}$ is the spectral vector, $k_{i,1}$ and $k_{i,2}$ are frequency indices and θ_i is a threshold.

It is clear, that the number of all possible combinations of indices is

$$N_\Phi = N_F(N_F - 1). \tag{3.2}$$

According to [23], we can use only a small subset out of N_Φ to build a classifier, because other combinations are not so speaker-specific.

Theoretically, the number of all possible features is infinite, because of the real threshold. In practice, we are limited by floating point precision and input data quantization, but still, the number of all possible features is huge. Thus, we must use a heuristic described in the footnotes for Section 3.2.2, when extracting features.

From Equation 3.1, we can see, that given some data, the feature can make decisions and predict one of two classes. In our speaker recognition case – client or impostor. This is also a nice example of how weak classifier can look for boosting purposes. If we combine several such features (weak classifiers) into one classifier, we can create a strong classifier. This is called *boosting* (see chapter 2.4).

The strong classifier is defined as a linear combination of such features:

$$F(\mathbf{x}) = \sum_{i=1}^{N_{\Phi^*}} \alpha_i f_i(\mathbf{x}), \tag{3.3}$$

where N_{Φ^*} is the number of selected features. This summation then serves as a speaker model. Here we see, that feature extraction and speaker modeling are very tightly linked.

The final decision is defined as:

$$\Omega^* = \begin{cases} \Omega_1(\text{client}) & \text{if } F(\mathbf{x}) \geq \Theta \\ \Omega_0(\text{impostor}) & \text{otherwise} \end{cases}, \tag{3.4}$$

where Θ is the detection threshold set correspondingly to a requested operating point of the system (see Section 2.5.1).

Important thing to note here is, that every speaker has a different subset of features. The question is, how to select good features for a client out of a complete (very large) set?

3.2.2 Extraction

We have defined the features, but we did not mention yet, how we can extract the useful ones and how to get concrete parameter values, for example the thresholds. This is described in the following text.

Let's define a function, that will measure the misclassification loss of some combination of features $F(\mathbf{X})$ on some training data $\mathbf{X} = \{\mathbf{x_j}\}_{j=1}^{N_{tr}}$ [23].

$$\mathcal{L}_{misc,F} = \frac{1}{N_{tr}} \sum_{j=1}^{N_{tr}} 1_{F(\mathbf{x}_j)y(\mathbf{x}_j)<0} \ , \tag{3.5}$$

where $\mathbf{x}_j$ is the j-th spectral vector (sample) in the training data matrix and where label $y(\mathbf{x}_j) = 1$ if $\mathbf{x}_j$ belongs to target class and $y(\mathbf{x}_j) = -1$ otherwise. The function simply computes the total number of misclassification errors over all the training samples.

We want to select a subset F, that will minimize this error. It is not tractable to minimize this function directly. Instead, we can use Discrete AdaBoost algorithm, that will minimize the exponential loss. This exponential loss is an upper bound of the misclassification loss [23], so it is safe (guaranteed to converge, if there are weak classifiers available with misclassification error lower than 0.5).

Discrete AdaBoost algorithm with weighted re-sampling

What follows is a complete training algorithm taken from [23]:

Inputs: N_{tr} training samples, i.e. spectral vectors $\{\mathbf{x}_j\}_{j=1}^{N_{tr}}$ extracted from the training speech data, their corresponding class labels, $y_j \in \{-1\,(\text{impostor}), 1\,(\text{client})\}$, N_{Φ^*}, the number of features to be selected, N^*_{tr}, the number of training samples to be randomly selected at each iteration ($N^*_{tr} < N_{tr}$)[1].

Output: The sequence of selected best features and its weights $\{\phi^*_n, \alpha^*_n\}_{n=1}^{N_{\Phi^*}}$.

1. Initialize the sample weights $\{w_{1,j}\} \leftarrow \frac{1}{N_{tr}}$
2. Repeat for $n = 1, 2, \cdots N_{\Phi^*}$:
   (a) Normalize the sample weights, $w_{n,j} \leftarrow \frac{w_{n,j}}{\sum_{j'=1}^{N_{tr}} w_{n,j'}}$
   (b) Randomly sample a subset of N^*_{tr} training samples, according to the distribution $\{w_{n,j}\}$.
   (c) For each feature ϕ_i in Φ, set the threshold parameter θ_i to minimize misclassification error,
   $$e_i = \frac{1}{N^*_{tr}} \sum_{j=1}^{N^*_{tr}} 1_{\{\phi_i(\mathbf{x}_j) \neq y_j\}}$$
   over the sampled set[2].
   (d) Select the next best feature, $\phi^*_n = \phi^*_i$ where $i^* = \arg\min_i e_i$, i.e. select that feature with the lowest misclassification error on the current subset of training samples.

[1] A value of N_{tr^*} equal to 5% of N_{tr} was found to work well [23].

[2] The difference $\mathbf{x}_j(k_i, 1) - \mathbf{x}_j(k_i, 2)$ for each training sample $\mathbf{x}_j$ is taken as a candidate threshold value. Any value in between two consecutive such thresholds would not change the classification result and hence can be ignored The optimal threshold θ_i is chosen via a search over these candidate values [23].

(e) Set $\beta_n \leftarrow \frac{e_{i^*}}{1-e_{i^*}}$

(f) Set the weight of the selected feature,

$$\alpha_n = -\log \beta_n$$

(g) Update the sample weights,

$$w_{n+1,j} \leftarrow w_{n,j} \beta_n^{1\{\phi_n^*(\mathbf{x}_j)=y_j\}}$$

3. Normalize feature weights to sum to one, $\alpha_n \leftarrow \frac{\alpha_n}{\sum_{n'=1}^{N_{\Phi^*}} \alpha_{n'}}$

Algorithm explanation

Here is the previous algorithm rewritten, using natural language, with difficult operations made clear.

After initialization, we select a subset of samples N_{tr}^* out of the training data, according to their weights. In other words, we select samples, that were wrongly classified before.

Then we select the best performing feature (or we can say, train a classifier) on this subset. This is done by a search over Φ. Further, for each feature ϕ_i we need to set the best possible threshold.

After we have selected the best feature on a subset in step 2c, we compute the misclassification error over *all* the samples and re-weight them accordingly (step 2g). Samples with higher misclassification error will get more weight and are more likely to be selected in the next iteration. The next iterations will try to select other features to correct this error.

At the end of the iteration, we also add the best feature/classifier to the set of selected features with the *feature weight* related to its performance *on the subset*.

3.3 Open questions

In this section, we present some ideas we have encountered during our experiments. These are also the differences between the BBF system proposed in [23] and our own implementation.

3.3.1 Number of candidate thresholds

Feature extraction is very time consuming task for BBF. To speed-up the feature selection process, we should optimize step 2*c* from the selection algorithm (see Section 3.2.2). There, the best feature is found by a search over all possible features and all possible thresholds. If we lower the number of candidate thresholds in this step (i.e. subsample it), we could extract features faster, possibly without a significant performance loss. Intuitively, we can say, that using larger amount of possible thresholds leads to a quicker boosting convergence, because we use „stronger“ weak classifiers. Later, we will refer to this number as to N_{thd}.

If `candid_thds` is the available threshold set, than we use its subset (MATLAB notation):

```
% sort thresholds and make them unique
thds=unique(candid_thds);
% linearly subsample only N_thd final thresholds
if length(thds) > N_thd
    thds=thds(floor([1:length(thds)/N_thd:length(thds)]));
end
```

3.3.2 Input data dimension

Another question is: What input data dimension is optimal? The input dimension is the half of the number of points obtained from the N-point DFT when preprocessing the data. The question can be rewritten as: What resolution in frequency do we need?

It is clear, that we should bear in mind the sampling frequency of the signal F_s.

The sampling theorem says, that the upper bound of frequencies present in a sampled discrete signal sampled at sampling frequency F_s is $\frac{F_s}{2}$. Higher frequencies appears in a sampled signal aliased and they are therefore usually band-limited before sampling. Because of this, we can imagine the input sampled signal to contain frequencies up to $\frac{F_s}{2}$ Hz, that are divided into $\frac{N}{2}$ bins by the DFT (because the signal is real in time, the magnitude spectra is symmetric and thus we use only a half[3] of it).

The BBF method uses the difference of magnitude in two frequency points for its computations. The frequency band it takes into account for one point is

$$\lambda = \frac{F_s}{N}$$

For example, for $F_s = 16\,\text{kHz}$ and $N = 256$ the distance of 2 points in spectra is 62.5 Hz.

Intuitively, higher resolution leads to better accuracy, but the system can fail, if the speaker's fundamental frequency increases or decreases in time of more than a $\frac{\lambda}{2}$.

Increasing the resolution also results in a quadratic increase of the number of all possible features N_Φ, that is mainly responsible for extraction time.

There are 2 ways, how we can manipulate frequency resolution of the resulting spectrogram. The first option is to change the window length of the STFT. This will result also in the change of the time resolution (number of output frames). We denote this operation later as a *STFT window resize to N samples*.

The second option is to keep the window of the STFT the same and change the parameter of a function computing the STFT, that determines how many points we want to get on the output (this is often labeled as NFFT). We denote this operation later as a *NFFT change to N points*. This will not change the frequency resolution itself, but it will rather interpolate the correct output (the signal segments will be zero-padded or shortened to the length NFFT). Note, that we loose information in the case of shortening. In the case of zero-padding, we do not lose any information, just the DTFT frequency representation is sampled smoother, in theory [29].

3.3.3 Training data set size and imbalance

This question relates to a size of a background set in the training data. For the TIMIT experiments, we followed the division into client ('1') and background ('-1') examples as

[3] Precisely, if the N is even, we get $\frac{N}{2} + 1$ unique points and if the N is odd, we get $\frac{N+1}{2}$ unique points.

presented in [23]. Target to non-target ratio in this case is $\frac{8}{250} = 3.2\%$. Total duration of the training data before preprocessing is $(8 + 250) \times 3\,s$, that gives about 13 minutes of recording.

The question is, what size and target to non-target ratio is optimal for the AdaBoost to work properly? There are some propositions to modify the AdaBoost, so it does not suffer from class imbalance in the training set [14][8]. We do not make any changes to the BBF system in this sense, but rather, we are aware of it.

3.4 Complete system

The BBF system can be implemented like every other speaker verification system, due to its principally the same architecture. This means, that it consists of two parts: training and testing. See Fig. 3.1.

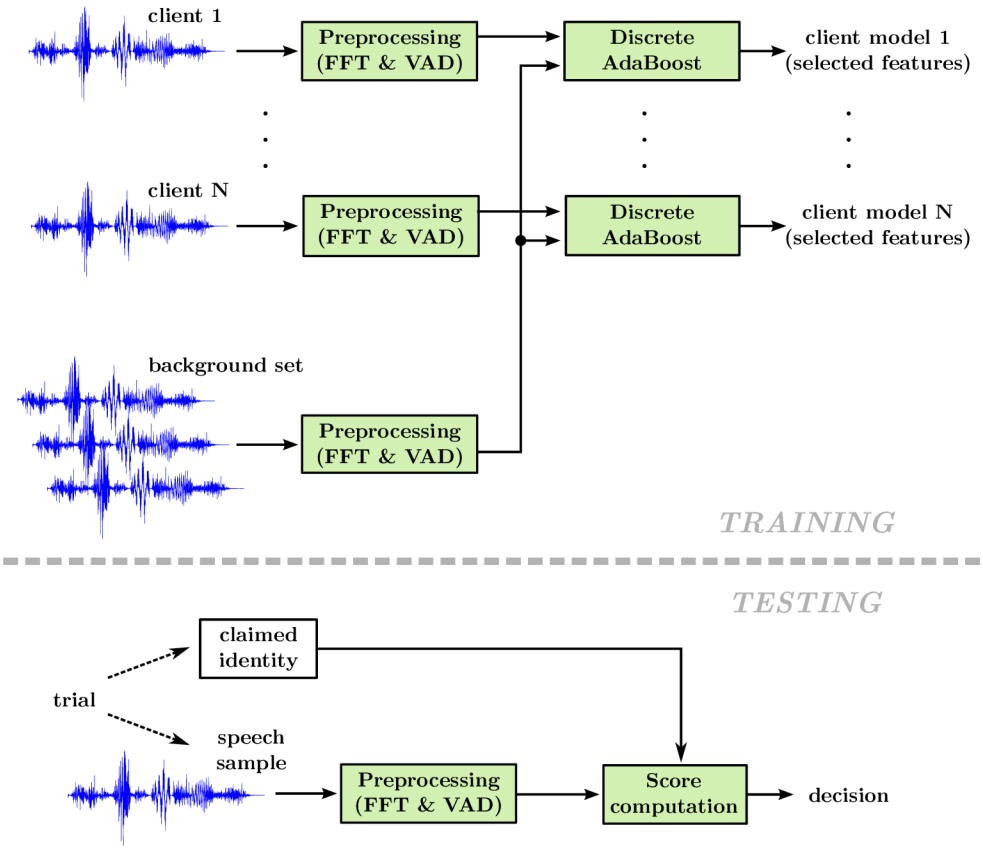


Figure 3.1: Overview of complete BBF recognition system

Note, that even if we use the term *background set* here, the system does not perform any kind of background model adaptation (like MAP adaptation for the GMM-UBM). Instead, the background set serves as a negative example for discriminative training by the AdaBoost algorithm. This means, that we have to train every client separately. If we used this system for a big set of speakers, we would need a lot of CPU time. But as the training is done offline, generally, it is not a handicap.

Chapter 4

Experimental setup and results

As a practical part of this work, we implemented the BBF system and tested it on two standard databases. The results are given in this chapter. Results for reference systems using classical feature extraction techniques are also given. Short discussion follows after each result and general topics are discussed in the next chapter, together with a summary of results.

4.1 Reference systems

In this section, we give a short description of recognition systems used later as a reference for our experiments on different data sets. For experiments on the TIMIT, we used own GMM-UBM system and for evaluation on the NIST SRE 2010 data set, we compare our BBF system with both the GMM-UBM and state-of-the-art iVector PLDA system. We kept the same preprocessing stage for both the BBF and the reference system, where possible.

4.1.1 Baseline GMM-UBM

The BBF system presented in [23] is compared to the results of Douglas Reynolds's GMM system from 1995 [22] (because they are better than results of their own, simpler, GMM implementation).

We compared our BBF system performance on the TIMIT with our own MFCC GMM-UBM, that uses the *same* background set of speakers and other common settings, such as the same VAD (Voice Activity Detector). We believe, that even if it is probably worse than the results of systems with speaker specific background speakers (*msc – maximally spread close* speakers [22]), the results are worth presenting.

Here, we indicate the most important values of system parameters: Diagonal covariance GMM contained 128 components. It was firstly trained as a UBM model containing the *same* background data (see Section 4.2.2) using the EM algorithm and then adapted into client models by MAP adaptation. All data was filtered through VAD and preprocessed into MFCC coefficients (20 coefficients were extracted). The score was computed as a ratio of client mixture and UBM mixture log likelihoods.

4.1.2 iVector PLDA system

We compare the BBF system performance on the NIST data with the state-of-the-art speaker recognition technology available in this work.

The full description of this system can be found in [5]. We give only an excerpt from [6] about this system here. The iVector is a low-dimensional, information rich, fixed length representation of an utterance. These low-dimensional data are then used as a „features“ for classifiers, the PLDA (Probabilistic Linear Discriminant Analysis) in this case.

iVectors used in this system had length 400 and were extracted via 60-dimensional features and 2048-component full-covariance UBM. The features were 19 short time gaussianized MFCC + energy augmented with their delta and double delta coefficients. The analysis window had 20 ms with shift of 10 ms. PLDA model was trained with 90 eigenvoices and 400 eigenchannels using mixer 04,05,06 and Switchboard telephone data [5].

The question is, how meaningful the comparison of the BBF and this reference system will be? The BBF training does not at all consider many problems, such as the channel variability, so we can a priori say, that this state-of-the-art system will be better. We do not try to compare the actual results, but we want to show by the results of this reference system, how far we can go with performance today.

At the beginning of this work, we also wanted to try out a score fusion, after the results of the BBF system on the NIST database have been available, but we had rejected to do so, after we have seen the actual results.

4.2 TIMIT

We repeated some experiments realized already in [23]. We kept the same testing set and evaluation conditions. In case we obtain the same results, we will be sure, that our system is properly implemented and the results given in the literature are correct (we will see, that they are).

Further, we used this relatively small database to experiment with some parameters of our system with extraction time in mind.

4.2.1 Data preprocessing

We should start with data preprocessing. Input signal is firstly transformed into frequency domain. According to [23] and originally according to [22], we use 20 ms sliding window over the input raw signal with a 10 ms shift to compute a 256-point DFT. We use one half of symmetric magnitude spectra.

After that, we use energy-based VAD (Voice Activity Detector) described in [15] to detect speech in signal and discard other frames without useful information. For our experiments, we used a distance of 30 dB from the maximum speech energy to set silence threshold. In [23], there is no remark about what VAD they are using, so we used simple energy based VAD described above. It is possible to use such simple detector, because the TIMIT contains clean and uniform data.

4.2.2 Setup

We followed evaluation protocol in [23] (first part originally described in [22]).

> The 168 speakers (112 males, 56 females) from the „test“ portion of the TIMIT database were used as clients. For each speaker, the 2 *sa* sentences, 3 *si* sentences and first 3 *sx* sentences were used for training and the remaining 2 *sx* sentences for testing.

...

For training a client classifier in the BBF system, i.e. to select the binary features using the AdaBoost algorithm, the positive (client, '1') training samples were extracted from the client training data, while the negative (impostor, '0') samples were extracted from a set of 250 utterances randomly selected from the „train“ portion of the TIMIT database (177 males and 73 females).

4.2.3 Results

The results that we obtained with our BBF system on the TIMIT are presented here. They are a bit worse than the results in [23], probably because we used an approximation described in Section 3.3.1 to speed-up computation. If not stated otherwise, we used $N_{thd} = 40$ for all computations (the value was chosen experimentally). This way, we could learn quickly about the BBF system behavior under other different conditions, without a need for a bigger computer.

We tried to evaluate our BBF system without N_{thd} limitation and the results were slightly worse than the results in [23]. This is probably because we used different VAD.

Input data dimension

As an answer to questions from the Section 3.3 (Open questions), we present the BBF system behavior for different input data dimensions in Fig. 4.1. The 20 ms window of the STFT was used for all experiments and we altered the frequency resolution by changing the NFFT parameter (see Section 3.3.2). This resulted in information loss, but the number of training frames for the AdaBoost remained the same. We give rough training times for these settings in section „Client model training time“.

As stated in Section 3.3.2, we can change the frequency resolution also without losing any information, by resizing the STFT sliding window. The results obtained with this preprocessing stage are in Fig. 4.2.

The reference GMM-UBM system (see Section 4.1.1) with the same preprocessing stage achieved for this task performance of 0.55% EER. This is better, compared to our best BBF system (with 512 features and using input data dimension of 128 and $N_{thd} = 40$), but note, that we used for this evaluation just an approximation of complete BBF system from [23] (see Section 3.3). With the full system (i.e. with system not limited by N_{thd}), they achieve EER of 0.31% on the same task.

Client model training time

This time depends mainly on input data dimensionality, then on the training data size, N_{Φ^*}, N_{thd} and of course on the CPU speed. Table 4.1 is just for a notion here. It is based on our MATLAB implementation that ran on the Intel® Core™ 2 Duo CPU E8200 @ 2.66GHz processor.

Compared to the reference GMM-UBM system, training is much more time consuming for the BBF system, because it can not use adaptation and every client has to be trained separately. It is like using the GMM for speaker modeling without the UBM MAP adaptation.

For an accurate analysis of computational complexity of the *testing* stage, compared to an evaluation of the GMM model, see section 5.6.2 in [23].

Input data dimension	**Feature set size (N_Φ)**	**Time to extract a feature [s]**	**Time total ($N_f = 512$) [min]**
32	992	1.53	14
64	4032	4.41	38
96	9120	10.65	91
128	16256	18.00	154

Table 4.1: BBF system training time

Client model size

We summarize some interesting facts related to speaker modeling here – client model size and in the next section client model training time. We show these properties to fully compare systems only – they are not the main objective today. The training is done offline and the disk storage is not an issue any longer.

For the BBF system, to achieve performance presented in this chapter, we need to know parameters for all 512 features. That is two integer frequency indices (4 B each), the threshold (8 B) and the feature weight (8 B). It is **12.228 kB** in total for the BBF system.

For the GMM-UBM system modeling, we need to store component weights (128), means (128×20) and covariance matrices (128×20). It is **41.984 kB** in total for the GMM-UBM[1].

Number of candidate thresholds

As we note in Section 3.3, throughout our work, we do not use the whole candidate threshold set for feature selection in our experiments. To speed-up computation, we use only a subset of it. This yields a question, how big error we suffer with given N_{thd}? Intuitively, we can say, that using larger amount of possible thresholds leads to a quicker boosting convergence, because we use „stronger“ weak classifiers. So the other interesting question is, how much this convergence relates to N_{thd}. We performed several evaluations concerning this question, so we could exploit this knowledge in the future. The results are shown in Fig. 4.3 and 4.4.

As we can see from Fig. 4.3, surprisingly, the actual value of N_{thd} has a minor effect on system accuracy. The difference in accuracy for $N_{thd} = 10$ and the full set (about 2000 thresholds on average) seems to disappear after we have extracted enough features.

The next question arises: How is this possible? We think, that even we get much smaller number of possible weak classifiers by lowering N_{thd}, we still get a bunch of good classifiers usable (imagine selecting a maximum value from a set of 1000 numbers with a Gaussian distribution and from a set of 100000 numbers with the same distribution – you will get very similar results).

This is an important information, because we can set N_{thd} to some small value and rapidly accelerate the feature selection process, without a significant performance loss.

We show results for a very low values of N_{thd} in Fig. 4.4. We can see, that by lowering N_{thd} to very low values in order of units, the accuracy drops (boosting convergence decelerates).

Note that this topic surely deserves more investigation.

[1] If we used only 32 mixture components (thus lowering the client model disk footprint to be equal that of BBF system → 10.624 kB), we would obtain performance EER=1.19 %.

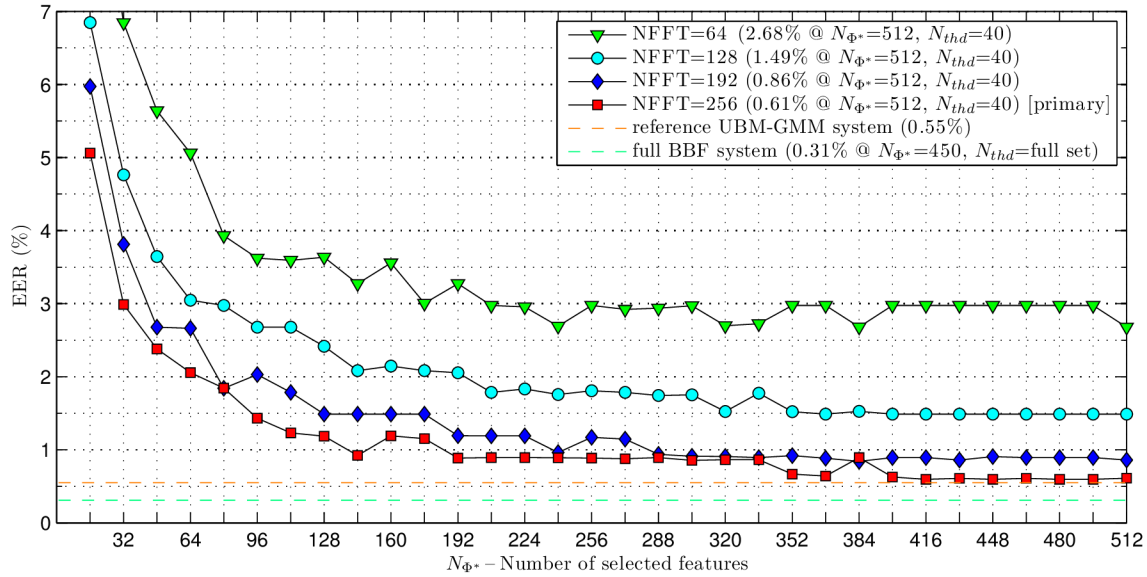


Figure 4.1: BBF system performance with respect to input data dimensionality and number of selected features N_{Φ^*}. Dimensionality was altered (with information loss) by changing the NFFT parameter only, with the same window size of 20 ms (320 samples). The results for *full BBF system* are taken from [23].

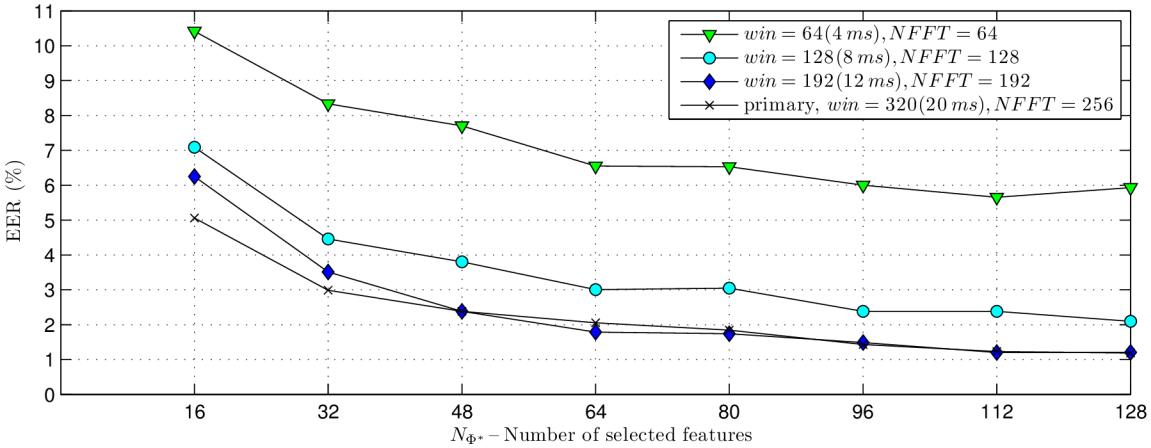


Figure 4.2: BBF system performance with respect to input data dimensionality and number of selected features N_{Φ^*}. Dimensionality was altered (without information loss) by changing the STFT window size. The last curve is our primary BBF system for reference, where the window was 20 ms (320 samples) and NFFT=256.

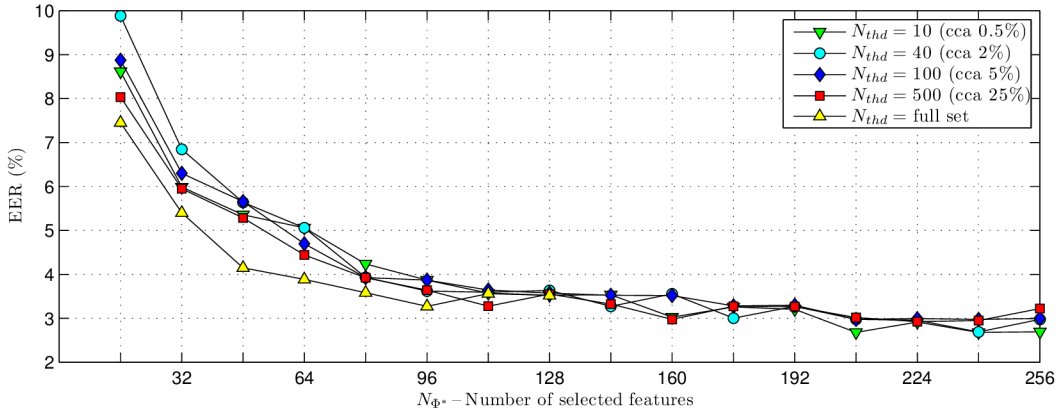


Figure 4.3: BBF system performance with different values of N_{thd}. For these experiments, we set NFFT to 64. The full set variant was evaluated only to $N_{\Phi^*} = 128$.

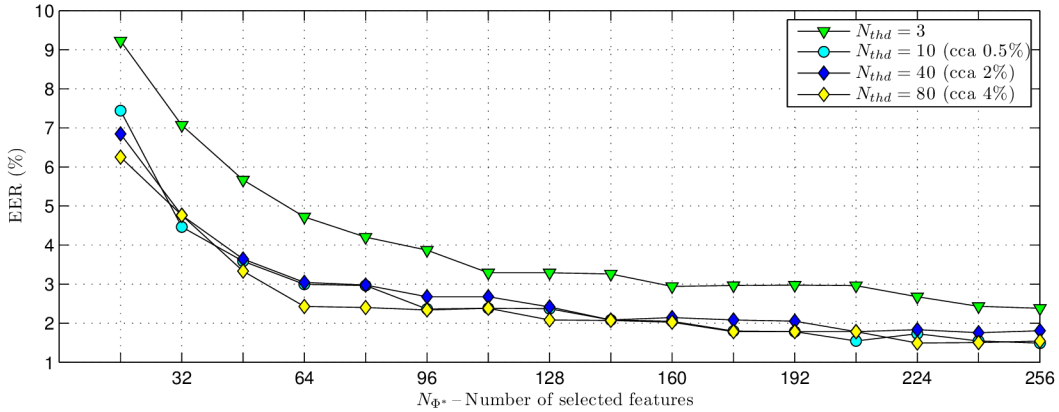


Figure 4.4: BBF system performance with low values of N_{thd}. The NFFT was set to 128 in this case.

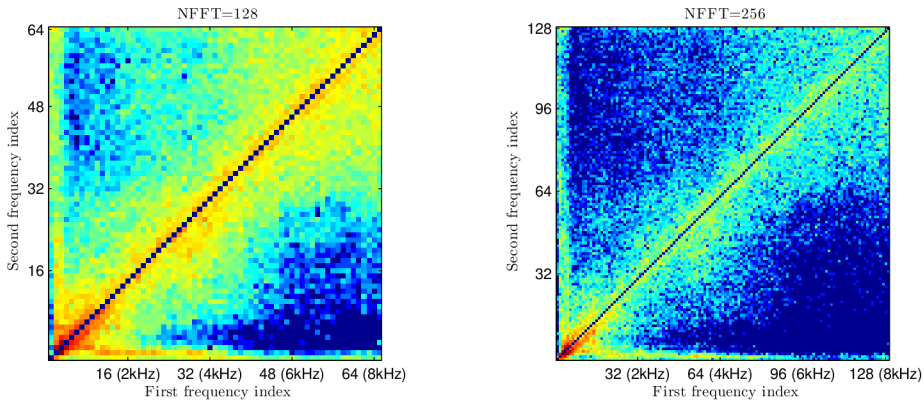


Figure 4.5: Distribution of selected features for NFFT=128 and NFFT=256. Features of all clients were pooled and counted for every frequency pair. Logarithm of this number was taken.

Other results

The feature distribution, as an interesting fact introduced in [24], is presented in Fig. 4.5. In this figure we can see, how many features were selected for a given point in the Φ-space (there are two coordinates in Φ-space – the first and the second frequency index). There were 512 features extracted for each client. Two variants are presented for different NFFT settings.

Note, that this is not a distribution of feature *weights* as in [23] on p. 79. In our case, the resulting weight distribution matrix (see Fig. 4.6) is rather flat, containing several (168) points with high average weight, that corresponds to the first selected features for each client. The weight distribution from [23] differs, probably due to different computing procedure.

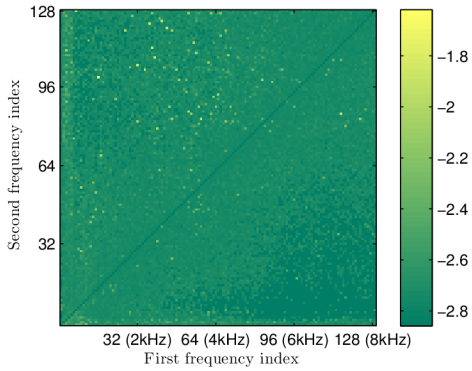


Figure 4.6: Distribution of selected feature weights. Features of all clients were pooled and expected feature weights for a given frequency pair were computed. log_{10} of this number was taken.

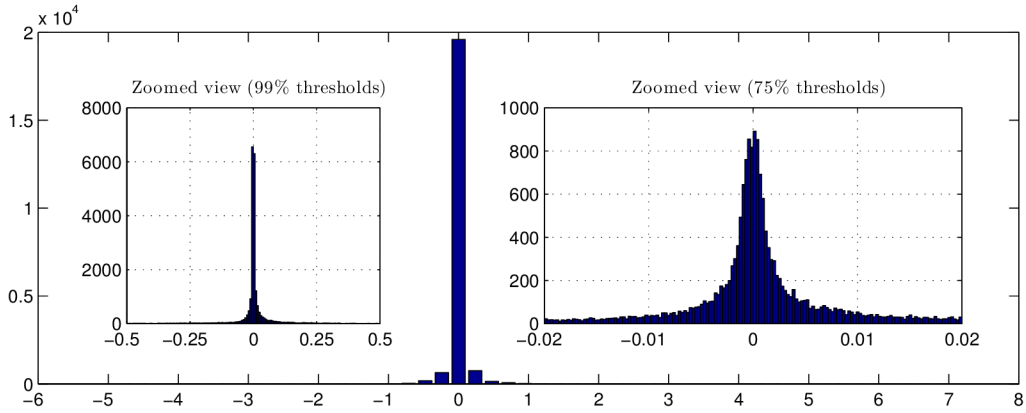


Figure 4.7: Histogram of thresholds of selected features. Individual thresholds ranges from -5 to 7 in the x-axis, but they are too sparse to be seen in the main histogram.

There is also a distribution (a histogram) of thresholds of selected features in Fig. 4.7. The number of candidate thresholds was not limited by N_{thd} in this case[2].

We can see, that most of the threshold values lies somewhere around zero. This is because the input data magnitude has a large dynamic range. We should mention the statistics of input data to compare threshold values with, now. The world set (background model) had maximum, minimum, mean and standard deviation equal to 13.9, 2×10^{-8}, 0.06 and 0.14 respectively. Parameter settings of feature extractor for this particular results were: N_{thd} = full set, $NFFT = 64$, $window = 20\,ms$.

4.3 NIST SRE 2010

As proposed in [23], we performed evaluation of the BBF system on the NIST SRE database. To our knowledge, the BBF speaker recognition system has been tested so far on clean speech (TIMIT), synthetic noisy databases (NTIMIT, HTIMIT and XM2VTS & Noisex-92, where each evaluation used *clean* speech for training) and the MOBIO database. The MOBIO database was created as a part of the Mobile Biometry (MOBIO) project [3]. The MOBIO Phase I database consists of speech data collected at 6 different sites from 152 people using a Nokia N93i mobile phone [23].

In this section we present the results we have obtained on the NIST database.

4.3.1 Experimental setup

All the input data was sampled at 8 kHz, so we decided to use 128-point Fourier transform and 20 ms window. The shift was set to a half of the window. As a consequence of what we have written in Section 3.3.2 about frequency resolution, the frequency resolution was the same as for experiments on TIMIT.

The BBF system was the same as in previous experiments. We also tried a different VAD in the preprocessing stage – phoneme recognizer-based segmentation available from the BUT speech research group *speech@fit*. Description of this VAD can be found in [5]. In short, it is a phoneme recognizer, where all phoneme classes are linked to the *speech* class. After this labeling, it uses energy based post-processing.

We used the female subset of the *core – condition 5* subset from the NIST SRE 2010 as a training data (one two-channel telephone conversational excerpt, of approximately five minutes total duration [19]). We chose the female subset, because females should be generally harder to recognize [6].

A subset of N_{BG} female speakers from the NIST SRE 2008 *short2* training condition was used as the background set.

After we had obtained very poor results on a preliminary subset of 128 speakers, we tried to alter some system variables, but with no luck. Table 4.2 describes these variables and their symbols, that are used later in results.

4.3.2 Results

The results, that were produced on preliminary subset of 128 speakers (with all modifications we have tried), are summarized in Table 4.3. The results of our BBF implementation,

[2] Although, when limited to $N_{thd} = 40$, the results were similar.

[3] http://www.mobioproject.org

Symbol	Description
N_{thd}	Number of candidate thresholds. (see Sec. 3.3.1)
N_{bg}	Number of background speakers. (see Sec. 3.3.3)
$N_{bgframes}$	Number of frames used for each background speaker. We used $N_{bgframes}$ to subsample background speaker data to ensure speaker variability and reasonable amount of data.
xnorm	Normalize input signal by its absolute maximum before preprocessing. We tried to normalize different speech levels on the input by setting the *xnorm* system parameter.
VAD n	This indicates that the simple energy-based VAD was used with silence to maximum energy distance n dB. Otherwise, phoneme based VAD described in 4.3.1 was used.

Table 4.2: The BBF system parameters for NIST database preliminary evaluation

in comparison with state-of-the-art systems, are very bad. The reference state-of-the-art system achieved EER of 3.57% [6] on the whole female subset of the NIST SRE 2010 condition 5. On the other hand, simple GMM-UBM system, as used for TIMIT experiments, performed similarly to our BBF system.

Because we did not find a way, how to improve the system performance on this subset at least to some reasonable level, we did not attempt to get results from the whole female subset.

As we can see from Table 4.3, the most influential variable for these experiments was VAD. The more benevolent we were with the distance of maximum signal energy to silence, the more the system performance decreased. This is clear, but with the settings, that has shown to be the best in our experiments (energy-based VAD with threshold 15 dB), we use only a fraction of available speech (only some vowels probably, because they have higher energies) and just lose valuable information this way. On the other hand, if we used more frames, the results were not better, perhaps because of added noise. Behavior of different VAD settings that we have used is demonstrated on a short utterance from the training set in Fig. 4.8.

We tried to partition the training data to be the same size and balance as the data from TIMIT from previous chapter (24 s of speech for client enrollment and target to non-target data ratio of 3.2%). This case is labeled as „timitlike“ in the results.

Although the output scores of BBF strong classifiers should be normalized on its own (as a consequence of Eq. 3.1), we present some z-normalized scores of our BBF system in the results. These are labeled as *z-norm* in Table 4.3. Normalization coefficients were estimated from a set of 32 cohort speakers, and cohort set was the same for all the clients.

4.3.3 Feature weight masking

The distribution of features selected for the B tests (see Table 4.3) is shown together with feature distribution obtained on TIMIT in Fig. 4.9. There are some new noticeable regions with high occurrence of selected features. These regions are encircled. We assume, that these are regions of *channel features* and *noise features*.

One could propose to use „correct“ feature distribution obtained on the TIMIT clean database to re-weight feature weights of features extracted from unclean NIST utterances. This way, it would be possible to eliminate the influence of channel and noise features,

i.e. the features, that were selected unintentionally by the AdaBoost on unclean training data. Likewise, this „masking“ would amplify weights of features, that are in regions of the Φ-space, where there were a lot of features selected for speakers on clean TIMIT speech. It could be thought as something like „feature-level noise/channel-effect cancellation“.

The question is, if it is possible to just re-weight non-linearly the feature weights and if this would be beneficial. We leave this topic open in this work.

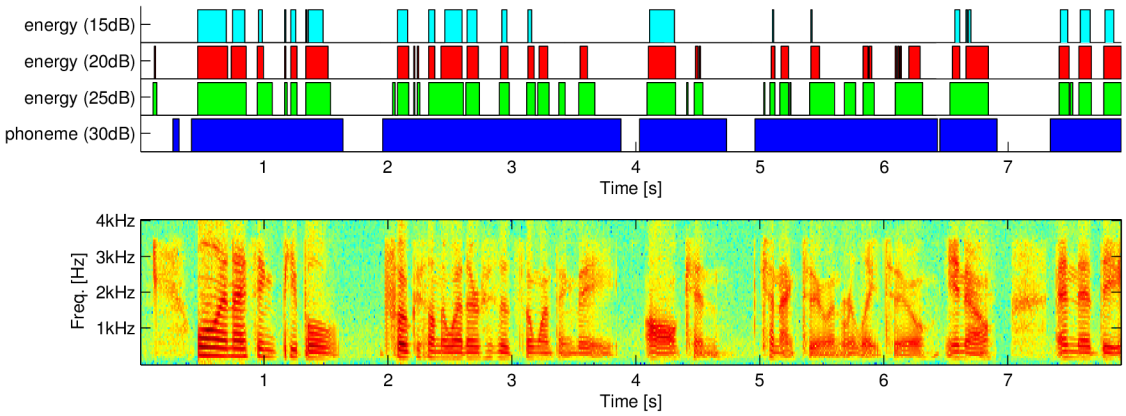


Figure 4.8: Different voice activity detectors we used for the NIST dataset. The first three are simple energy-based detectors with different thresholds and the third one is phoneme-based detector described in Section 4.3.1

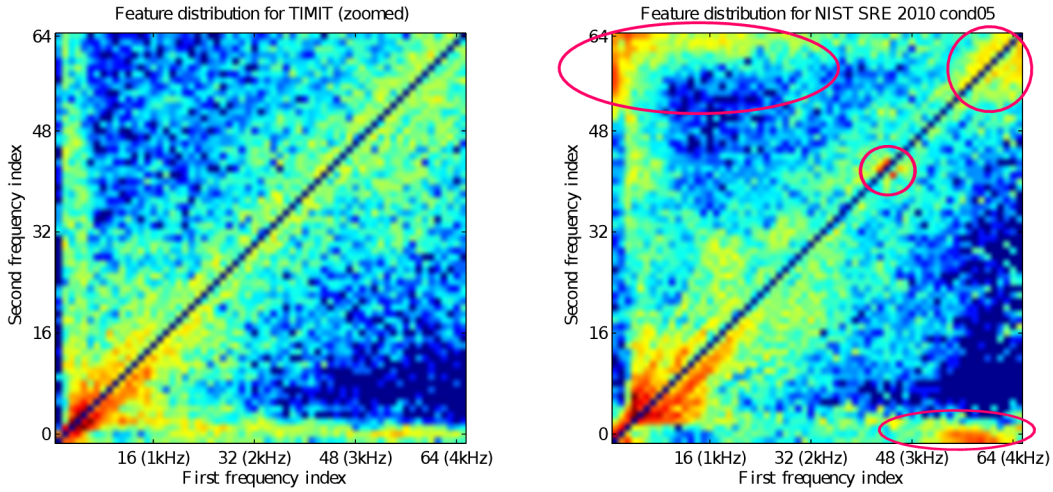


Figure 4.9: Distribution of features extracted on clean speech (on the left, TIMIT, males and females) and on noisy training data with channel variability (on the right, NIST SRE 2010, telephone speech, females). The main differences are highlighted.

test group	**% EER**	**VAD**	$N_{\Phi*}$	N_{bg}	$N_{bgframes}$	N_{thd}	**xnorm**	**additional**
	36.03	15	64	400	1000	40	yes	
A	35.67	15	128	400	1000	40		z-norm
	35.22	15	128	400	1000	40		
	38.04	20	32	300	800	20	yes	
	37.51	20	64	300	800	20	yes	
	36.73	20	96	300	800	20	yes	
	36.44	20	128	300	800	20	yes	
	36.71	20	160	300	800	20	yes	
B	36.20	20	192	300	800	20	yes	
	35.86	20	256	300	800	20	yes	
	36.08	20	288	300	800	20	yes	
	36.27	20	320	300	800	20	yes	
	35.95	20	352	300	800	20	yes	
	36.08	20	384	300	800	20	yes	
	37.67	20	384	300	800	20	yes	z-norm
	39.82	phn	64	300	1000	40		
	40.14	phn	64	250	1000	40	yes	
C	40.41	phn	128	250	1000	40	yes	
	40.40	phn	192	250	1000	40	yes	
	40.04	phn	256	250	1000	40	yes	
	39.99	phn	32	250	1000	20	yes	WIN=NFFT=128
D	40.32	phn	64	250	1000	20	yes	WIN=NFFT=128
	39.70	phn	96	250	1000	20	yes	WIN=NFFT=128
	39.65	phn	128	250	1000	20	yes	WIN=NFFT=128
	41.03	phn	32	250	1000	20		WIN=NFFT=128
E	39.71	phn	64	250	1000	20		WIN=NFFT=128
	39.80	phn	96	250	1000	20		WIN=NFFT=128
	41.95	phn	128	250	300	80		timitlike
F	40.41	phn	259	250	300	80		timitlike
	41.35	phn	323	250	300	80		timitlike
iVec+PLDA reference	3.57	phn	-	-	-	-	-	
GMM-UBM reference	36.30	15	-	250	1000	-	yes	

Table 4.3: The BBF system performance for different settings of parameters on the NIST SRE 2010 subset of 128 females. Reference systems are described in Section 4.1.2. iVec+PLDA reference system was evaluated on the *full* female subset.
Test group labels A-F are there for later referring and they have no other meaning.

4.4 Results and discussion

Our implementation of the BBF system was evaluated on two different databases. Let's recapitulate all the results in this section.

4.4.1 Experiments on the TIMIT

From the results obtained on the TIMIT database, we can see that the BBF system really is comparable with classical approaches and feature extraction techniques (MFCC GMM-UBM in this case) on clean speech. It has been shown in Sec. 4.2, that both systems have its own pros & cons and all these properties should be considered before looking at the EERs we give, because they are really not an absolute measure (we could use 512 mixture components in the GMM, instead of 128, for example).

As an interesting fact, we would like to highlight the result from Sec. 4.2.3, i.e. the finding, that we do not have to consider the whole set of candidate thresholds, when selecting the best feature in each AdaBoost iteration, and thus speed-up the feature extraction process.

4.4.2 Experiments on the NIST SRE data

The results obtained on the NIST SRE 2010 core condition 5 were not as good because the training data was much harder. We try to discuss this problem here. We think that it has several reasons.

1. The BBF system does not consider channel effects at all, but the training contains this variability. So the AdaBoost probably selected also a „channel features" rather than characteristic features of speakers. Phones and channels used to record speech in the background set were different to enrollment phones and channels. This is a big mismatch. The particular selection of the background set could be also the reason.

2. The amount of the training data and noise presence caused the AdaBoost to over-fit the noisy frames. The AdaBoost noise over-fitting problem is known already [7][9] and so the next logical step could be to try out to incorporate some more robust boosting algorithm. We suggest concrete directions in the following chapter.

3. The training set was imbalanced. This was not investigated further in this work. Maybe it is not a problem at all.

Although the BBF system was much worse, than the state-of-the-art reference system, simple GMM-UBM system achieved similar results. This is because these simple systems do not consider channel effects and noise. It is interesting, from our point of view, that both BBF and GMM-UBM again achieved similar results.

Chapter 5

Conclusions and future work

The BBF technique for feature extraction and speaker modeling was investigated in this work. We confirm the results given in the original paper [23] and we present some new results, ideas and propositions. As proposed in [23], the BBF system was evaluated on the NIST SRE data, which was also the main goal of this work – objectively compare classical feature extraction techniques with the system based on computer graphics features.

The results of this evaluation are not very optimistic, but this could be expected, due to the simplicity of the classifier and difficult evaluation data. Performance of simple GMM-UBM and our BBF system was similar, which means, again, that the MFCC and BBF features are comparable. The main problem with BBF is, that there are no training methods available so far, that consider channel/speaker variability and/or noise in training data. So, it will be always worse, than advanced systems with this knowledge, on such data. This does not mean, that the BBF system is not worthy of further interest. There are applications and scenarios, like mobile biometry, where it is proven to work finely and with many advantages over other methods (simplicity, quick score computation, robustness to unknown noise). Well, it was designed for this task.

5.1 Future work

There are many possibilities, how to continue on this topic. During our work, we have encountered several problems. We think that these problems deserves more attention and time, we had not had.

We briefly touched the topic of phonetic class-specific modeling (something similar to phoneme-specific GMM from [13]), that we have not documented in this paper. The idea behind this is, that speaker features extracted from, for example vowels, uses totally different frequency regions, opposed to consonants. If we have divided speech into these classes before training and testing, we could benefit from better accuracy.

Previous direction was short term, i.e. straightforward and quick to try out and test. What would be more complicated, but very interesting, is to exploit the information about feature distribution (see Fig. 4.5) in some way. In Section 4.3.3, we propose an idea, how to enhance features extracted from unclean data by re-weighting their feature weights. This is an example of such exploit.

It could be useful, to somehow improve the boosting process itself, for example using the Totally Corrective AdaBoost [28] or incorporate boosting algorithms more appropriate for noisy data like the BrownBoost [9].

Bibliography

[1] Yali Amit, Alexey Koloydenko, and Partha Niyogi. Robust acoustic object detection. *Journal of the Acoustical Society of America*, 118:2634–2648, 2005.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[3] Frédéric Bimbot, Jean-François Bonastre, Corinne Fredouille, Guillaume Gravier, Ivan Magrin-Chagnolleau, Sylvain Meignier, Téva Merlin, Javier Ortega-Garcia, Dijana Petrovska-Delacrétaz, and Douglas A. Reynolds. A tutorial on text-independent speaker verification. *EURASIP J. Adv. Sig. Proc.*, 2004(4):430–451, 2004.

[4] Jake V. Bouvrie, Tony Ezzat, and Tomaso Poggio. Localized spectro-temporal cepstral analysis of speech. In *Proc. of ICASSP*, pages 4733–4736. IEEE, 2008.

[5] Niko Brummer, Lukáš Burget, Patrick Kenny, Pavel Matějka, Edward Villiers de, Martin Karafiát, Marcel Kockmann, Ondřej Glembek, Oldřich Plchot, Doris Baum, and Mohammed Senoussauoi. ABC system description for NIST SRE 2010. In *Proc. of NIST 2010 Speaker Recognition Evaluation*, pages 1–20. National Institute of Standards and Technology, 2010.

[6] 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.

[7] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–157, 2000.

[8] Wei Fan and Salvatore J. Stolfo. AdaCost: misclassification cost-sensitive boosting. In *Proc. of 16th International Conf. on Machine Learning*, pages 97–105. Morgan Kaufmann, 1999.

[9] Yoav Freund. An adaptive version of the boost by majority algorithm. In *Proc. of the Twelfth Annual Conference on Computational Learning Theory*, pages 102–113, 2000.

[10] Yoav Freund. Linear separation, drifting games & boosting. [online], 2009. Video lecture available at URL: `http://videolectures.net/icml09_freund_dgb/`.

[11] Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 1999.

[12] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[13] Eric G. Hansen, Raymond E. Slyh, and Timothy R. Anderson. Speaker recognition using phoneme-specific GMMs. In *Proc. of Speaker Odyssey: the Speaker Recognition Workshop (Odyssey 2004) (Toledo Spain, May 2004)*, pages 179–184, 2004.

[14] Mahesh V. Joshi, Vipin Kumar, and Ramesh C. Agarwal. Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *Proc. of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 257–264, Washington, DC, USA, 2001. IEEE Computer Society.

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

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

[17] Hong C. Leung and Victor W. Zue. Visual characterization of speech spectrograms. In *Proc. of ICASSP*, pages 2751–2754, 1986.

[18] Prachi J. Natu, Shachi J. Natu, Dr. T. K. Sarode, and Dr. H. B. Kekre. Performance comparison of speaker identification using DCT, Walsh, Haar on full and row mean of spectrogram. *International Journal of Computer Applications*, 5(6):30–37, August 2010. Published By Foundation of Computer Science.

[19] National Institute of Standards and Technology. *The NIST Year 2010 Speaker Recognition Evaluation Plan.* NIST, 2010.

[20] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29:51–59, 1996.

[21] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, page 1–8, 2007.

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

[23] Anindya Roy. *Boosting Localized Features for Speaker and Speech Recognition.* PhD thesis, École Polytechnique Fédérale de Lausanne, 2011.

[24] Anindya Roy, M. Magimai-Doss, and S. Marcel. Boosted binary features for noise-robust speaker verification. In *Proc. of ICASSP*, pages 4442–4445, 2010.

[25] Anindya Roy, Mathew Magimai-Doss, and S. Marcel. Phoneme recognition using boosted binary features. In *Proc. of ICASSP*, pages 4868–4871, 2011.

[26] Anindya Roy, Mathew Magimai.-Doss, and Sébastien Marcel. A fast parts-based approach to speaker verification using boosted slice classifiers. *IEEE Transactions on Information Forensics and Security*, 2011.

[27] Paul A. Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, pages 511–518, 2001.

[28] Jan Šochman and Jiří Matas. Adaboost with totally corrective updates for fast face detection. In *Proc. of the Sixth IEEE international conference on Automatic face and gesture recognition*, FGR' 04, pages 445–450, Washington, DC, USA, 2004. IEEE Computer Society.

[29] Jan Černocký. Signals and systems: course materials. [online], 2009. URL: `https://www.fit.vutbr.cz/study/courses/ISS/public/.cs`.

Appendix A

Implementation details

We used the MATLAB® for all computations and BBF system implementation. All the source code, results, figures etc. is available on the attached CD-ROM.

A.1 Directory structure

Here we highlight only something from the CD-ROM contents (we omit the libraries and less important files):

`{TIMIT | sre10}_eval/`	TIMIT and NIST specific evaluation code
`scores/`	evaluation scores in BOSARIS friendly format
`bbf.m`	extractor of BBF features
`constants.m`	system parameters can be altered here
`*.sh`	Sun Grid Engine shell scripts for parallel task management
`lib/`	code for data preprocessing and the AdaBoost implementation
`tex/`	LaTeX source code for this paper
`figs/`	several .eps figures used in this text and scripts for its creation
`gmm-ubm/`	GMM-UBM implementation for TIMIT data
`ubm_sre10_eval/`	GMM-UBM implementation for NIST SRE data

A.2 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. Also, there is nothing like graphical front-end. Everything is done by calling scripts from command line. Here, we would like to provide a small programming guide, at least to allow others to repeat the experiments.

First thing to do is to set paths correctly. Paths are defined for different datasets in files `constants.m`. Location of the TIMIT database is defined in `TIMIT_eval/submit.sh` as an environment variable `TPATH`. This variable must be set before running any script from the `TIMIT_eval/` directory. Path definitions for NIST data should be altered in `sre10_eval/constants.m`.

The root directory for training outputs (extracted features and cache) is defined in the variable `savedir`. The outputs for different parameter settings are stored in subdirectories called after the `idstring` variable (this variable is created automatically from current settings).

The AdaBoost algorithm is implemented in two files: `lib/AB_train.m` and `lib/AB_step.m`. The main function `AB_train(data, labels, niter, savefile, nthd)` is called from the main training script `bbf.m`. If `savefile` exists, it is loaded and the training continues, where it has stopped previously (this can be altered by setting `continue_train` variable). The extracted features are being saved together with a big vector of *sample weights* to make this possible.

For TIMIT, you can use `eval_preproc.m` for score generation of actual parameter settings or `eval_NfDepend.m` to generate results like from Fig. 4.1. For the NIST database, use `score_subset.m` for score generation of actual parameter settings. The subset of 128 females we use in this work is defined in file `subset10c05f_key.txt`.

Scores can be processed using the BOSARIS Toolkit or the NIST evaluation tools.

We used the Sun Grid Engine (SGE) environment for parallel feature extraction. Shell script `submit.sh` can be used to submit array job to grid and train all the models defined in `clients_ids.txt` for TIMIT and in `subset10c05f_clients.txt` for NIST data.

There are some helper scripts that will output summary of feature extraction so far. `lib/desc_feas.m` should output a list of features already extracted in the format „idstring number-of-selected-features“. `ccat.m` outputs detailed information about features already selected for current parameter settings.

A.3 Libraries and toolkits used

This is a complete list of second party software used in our work. All is freely available on the Internet.

- **VOICEBOX**[1] – Speech Processing Toolbox for MATLAB
- **BOSARIS Toolkit**[2] – MATLAB code for calibrating, fusing and evaluating scores from (automatic) binary classifiers
- **Netlab**[3] – Netlab Neural Network Software from Aston University
- **MatlabADT**[4] – MATLAB Audio Database Toolkit enables easy access and filtering of audio databases such as TIMIT and YOHO by their metadata
- **NIST valuation tools**[5]
- **Useful Matlab Functions for Speaker Recognition Using Adapted GMM**[6] by Md Sahidullah (requires Netlab)
- **IKR course demos**[7]

[1] http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
[2] http://sites.google.com/site/bosaristoolkit/
[3] http://www1.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/
[4] http://www.mathworks.com/matlabcentral/fileexchange/23843-matlab-audio-database-toolbox
[5] http://www.itl.nist.gov/iad/mig//tools/
[6] http://www.mathworks.com/matlabcentral/fileexchange/31678-useful-matlab-functions-for-speaker-recognition-using-adapted-gaussian-mixture-model
[7] http://www.fit.vutbr.cz/study/courses/IKR/public/demos/