CENTER FOR
MACHINE PERCEPTION

CZECH TECHNICAL
UNIVERSITY

RESEARCH REPORT

ISSN 1213-2365

Detection of Unusual Acoustic Events

David Gomez Vicario

qqgomez@cmp.felk.cvut.cz

Polytechnical University of Catalonia

CTU–CMP–2007–19

September 5, 2007

Supervisor: Tomas Pajdla

This reasearch has been supported by FP6-IST-027787 project
DIRAC and MSM 6840770038

Research Reports of CMP, Czech Technical University in Prague, No. 19, 2007

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Detection of Unusual Acoustic Events

David Gómez Vicario

September 7, 2007

A mi abuelo, que no pudo ver esto.

A mis padres, Adolfo y Mercedes, por haberme dado esta oprtunidad y a mi hermano Carlos por estar siempre pendiente de mi.

A mis amigos de la infancia, que siguen y seguirán siempre a mi lado; Javi, Abel, Edu, Gema,... Gracias por todos esos momentos inolvidables.

Como no a Patri, por su cariño y afecto hacia mi. Gracias por iluminarme siempre el camino.

Aknowledgment

First of all, I would like to thank the support given by Tomas Pajdla, my master thesis supervisor. This report is a reality thanks to his experience, patience and understanding. Hynek Bakstein and Javier Hernando deserve also a special mention for their continuosus support.

I also want to thank all my university and erasmus friends, by all those good moments that will remain forever in my memory. Specially to Emilio, Alex, Andrés S., Andrés L., Xavi, Nacho, Roger, Juan D. G., Alberto, Sergio, Joan Antoni, David G., Javi, Marc, Luis G., Luis H., Albert, Miriam, Neus, Anna, Jesica, ... and others, not less important although unnamed.

A final acknowledgement goes for all those who know that deserve it, incluing the readers of the present report, for having the patience for reading it.

Contents

List of Figures

List of Tables

Abbreviations

AB	Ada Boosting
AMS	Amplitude Modulation Spectrogram
ASR	Automatic Speech Recognition System
BFS	Bark Frequency Scale
CV	Cross Validation
FC	Carrier Frequency
FFT	Fast Fourier Transform
FM	Modulation Frequency
FN	False Negatives
F0	Fundamental Frequency
FP	False Positives
FPS	Frames Per Second
FT	Fourier Transform
GUI	Graphical User Interface
LTFT	Long Time Fourier Transform
LPB	Linear Programming Boosting
MA	Manner of Articulation

MFS	Mel Frequency Scale
MFT	Mel Frequency Transformation
MLP	Multi Layer Perceptron
PA	Point of Articulation
PCA	Principal Components Analysis
ROC	Receiver Operating Characteristic
SFS	Sequential Forward Selection
SLI	Spoken Language Interface
SND	Speech/Non-speech Detection
SNR	Signal to Noise Ratio
SP	Sound Pressure
SPL	Sound Pressure Level
STFT	Short Time Fourier Transform
SVM	Support Vector Machine
TB	Total Boost
TOH	Threshold of Hearing
TOP	Threshold of Pain
TSSP	Time Sequence of Spectral Parameters

Chapter 1

Introduction

1.1 Motivation

One of the major problems in any spoken communication is the presence of noise or other acoustic events that may prevent a correct understanding of the message. Such fact takes place in many daily situations: a conversation in the street, a telephone call in a supermarket, etc. This problem has worse effects on machines or systems with a Spoken Language Interface, such as Automatic Speech Recognition systems (ASR).

One of the challenges in the speech recognition area is to provide "clean" speech streams to the SPL systems, so that it may result in greater recognition accuracy. This is one the purposes of the speech/non-speech detection, which, moreover, corresponds to the topic of the present work.

1.2 Objectives

The goal of the project is to review the state of the art of modeling and detection of unusual audio events, implement a state of the art technique, and test it on real data. In other words, we want to build a toolbox to identify some acoustic events.

In particular, we are interested in the speech/non-speech detection. That means that, given an audio signal, we would like to know if it corresponds to speech or to any other acoustic event (non-speech). A key step to do that is to select relevant features from the audio signal. Such features can be obtained by means of the psychoacustically motivated amplitude modulation

spectrogram (AMS) [12].

Because there are diverse methods to carry out the previous task, this work is oriented to observe a novel procedure's performance, based on the AdaBoost algorithm, which was formulated by Yoav Freund and Robert Schapire [6]. As any other boosting technique, AB builds a strong classifier by combining several weak learners. The difference, nevertheless, is that the new weak classifiers are trained to favor those instances misclassified by previous classifiers. For that reason, it is hoped to achieve a good accuracy in the acoustic detection, with a reasonable number of features or weak classifiers.

1.3 Related work

The starting point of the present study is the work presented by Schmidt and Anemüller [17]. The goal of such work is to determine the features that perform the best classification and provide a good generalization to new signals. The feature extraction is carried out by means of the AMS, whereas the feature selection is based on the support vector machine (SVM).

Feature selection is performed with a standard sequential forward selection algorithm (SFS), which identifies the best feature subset for a given number of features. Those are the features that achieve a higher classification accuracy. Then, to avoid overfitting, K-fold cross-validation method is used.

In order to tackle the problem, 3 experiments are carried out. The first one tries to determine the most salient modulation frequency bands. It is shown that the most salient one is the 3 Hz-band, followed by the 4, 26, 25, 9, 14, 28, 13, 20 Hz-bands. The second experiment tries to reduce the number of features by selecting the center frequencies, corresponding to the mentioned most salient f_m-bands, that really contribute to the classification accuracy. At the end of this point, the total number of classification features has decreased from 493 values to 54. Finally, the third experiment tries to observe the influence of the selected features on the generalization ability in new acoustic environments. By using the classifiers trained under the second experiment, only feature numbers over 50 perform the classification task decently.

As a conclusion, it is shown that the number of features needed for speech detection can be reduced by using AMS patterns, which does not entail

a significant loss of accuracy. Furthermore, the most salient modulation frequencies for the speech/non-speech classification task are determined.

Krishna, Motlicek and Gatica-Perez [13] is also based on the modulation spectrum as a feature extraction method to examine the frequency rates dominant in speech. Then, some classification techniques, such as short-term energy, short term-energy and zero-crossing based segmentation techniques, and the multi layer perceptron (MLP) classifier system, are used to tackle the problem.

The speech/non-speech detection (SND) is crucial in order to improve the speech recognition accuracy. Inacurate boundaries cause many errors in this kind of systems. The proposed approach is based on long-term modulations to examine the slow temporal evolution of the speech energy. The algorithm is designed according to one particular characteristic of speech, which says that the use of components from the range below 2 or above 16 Hz can degrade the recognition accuracy.

The speech recognition evaluations are performed for all the methods and varying some parameters such as, signal-to-noise ratio (SNR) or distant microphone. The results show that the proposed method can be applied for any mode of speech acquisition and unforeseen conditions.

1.4 About this document

The following report is written to be a reference for the SND task. This means that all the necessary knowledge to understand the entire process are included. To make it simpler, it is organized in three main parts. The first one consists in the necessary background theory to understand the work which is described next. It is based on the documentation found about the current topic, related work and other complementary material. Acoustics, speech processing and pattern recognition are some of the discussed knowledge. The second part is the work itself. Here, all used procedures and algorithms are detailed, as well as the chosen values for their main parameters. The last part is the experimental and discussion part. On the appendix, there is additional information about the results achieved during the classification task, as well as information about the program used to that end.

Part I

BACKGROUND THEORY

Chapter 2

Nature of sounds and speech

In this section, we review some basic concepts about speech and sound in general. We will also define production, perception and main features of both. This information is based on [10] and some other sources that will be cited at the appropriate moment.

2.1 Sounds

Sound is a longitudinal pressure wave produced by a vibrating object that travels through a medium and that can be characterized by the general properties of waves, which are amplitude, frequency, period, wavelength and speed. It can be perceived by the sense of hearing. Air, water or earth are examples of such media.

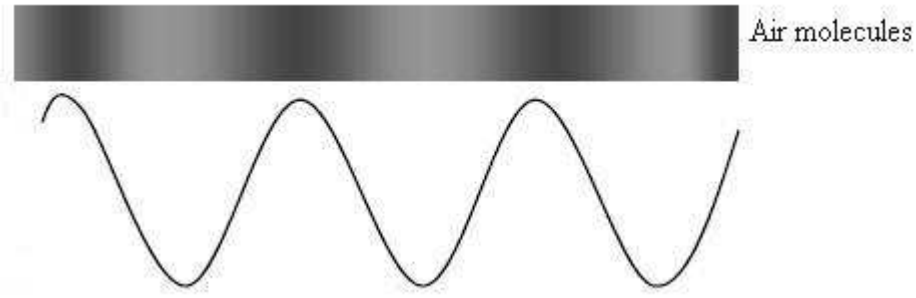

Figure 2.1: Compressions and rarefactions are produced by the application of the energy and can be described by a sine graph.

As shown in Figure 2.1, This pressure wave is formed of compressions and rarefactions of air molecules, following a parallel direction to that of the application of energy. Compressions are zones with high density of air molecules. On the other hand, rarefactions are zones with low density. The alternation between compressions and rarefactions of air molecules is sometimes described by the graph of a sine wave, as shown below. Crests of the sine correspond to compressions and troughs to rarefactions.

The use of a sine curve doesn't mean that sound waves have this shape, because air particles are oscillating at the same place. Its purpose is only to indicate pressure variations over time. The speed of a sound pressure wave depends on the medium through which the wave is crossing. In the air it is approximately 340 m/s.

2.1.1 Perception of sound

The ear is the part of the body of animals and humans meant to hear. Thus, perception is accomplished by the sense of hearing. Sounds can be used for communication or to get information about the surrounding environment.

The audible range includes the frequencies within the range 20 Hz to 20 kHz, so out of this, no sounds can be heard by the human ear. However, this range depends on each person, on the gender and on the age. Most human speech communication takes place between 200 and 8,000 Hz and the human ear is most sensitive to sounds from 1000 to 3,500 Hz (telephonic channel).

The amplitude of a sound is usually given in pressure terms and because of the wide range of amplitudes that a human ear can detect, a logarithmic decibel amplitude scale is used. The lowest sound amplitude perceptible by the ear is about 20 μPa.

2.1.2 Sound pressure

Sound pressure measures the difference between the local ambient pressure, with no sound, and the local ambient pressure in presence of a sound wave.

Due to the wide range of sound amplitudes that the ear is able to detect, it is convenient to measure them on a logarithmic scale in *decibels*. Such a scale is useful to compare two sounds:

$$10 \log_{10} \left(\frac{P_1}{P_2} \right)$$

where $P_1$and $P_2$are the two power levels.

On the other hand, sound pressure level (SPL) gives us information about the absolute pressure level and it can be defined, in dB, as:

$$SPL(dB) = 20 \log_{10} \left(\frac{P}{P_0} \right)$$

where the reference pressure, P_0, is 0 dB and corresponds to the threshold of hearing (TOH), which is the faintest audible sound. Its value is $P_0 = 20\mu Pa$ for a tone of 1kHz. As a curiosity, the speech conversation level has a SPL about 60 dB and the loudest sounds tolerated by the human ear are about 120 dB. At Table 2.1. we can see these and other values for common sounds.

Sound	**Sound pressure (Pa)**	**SPL (dB)**
Threshold of hearing	0.000020	0
Whisper	0.00006325	10
Normal conversation	0.02	60
Rock music	11.25	115
Threshold of pain	20	120
Rupture of eardrum	2000	160

Table 2.1: Absolute and relative pressures for common sounds

2.2 Speech

Spoken language is the means by which a speaker can communicate to a listener. It can be used to express feelings, emotions, ideas, etc. Speech production and perception are the main tasks of an oral communication. Speech production begins with a thought produced in the brain of the speaker and with the aim to communicate something. It is in fact, the brain, which origins the necessary movements in the muscles required to produce sounds. On the other hand, a listener receive a sound wave through the ear and with

the help of the auditory system transforms it into the neurological signals that the brain can understand.

Speech signals are composed of analog sound patterns that will be used later to achieve our main goal: to distinguish speech from non-speech.

2.2.1 Speech production

We review here the basic concepts, which are used for speech recognition and speech synthesis systems.

2.2.1.1 The vocal tract

Speech is produced by air-pressure waves coming from the mouth and the nose of the speaker. After these waves have been generated at a sound source called larynx, they go through the vocal tract , which is a cavity where the sound is filtered [15].

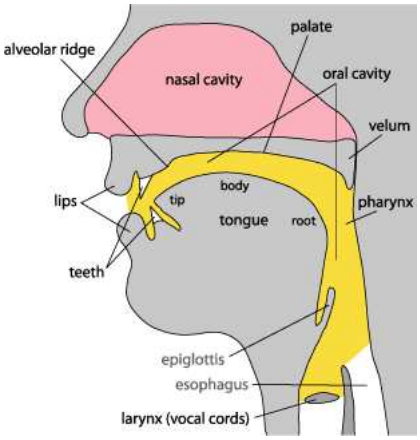

Figure 2.2: The vocal tract

There are two basic kind of sounds:

- consonants - on its production, the air way out is totally or partially obstructed by the different articulators

- vowels - articulated without major constrictions and obstructions. The air goes out free.

There are a wide variety of consonant sounds according to the manner (MA) and the place of articulation (PA). The first one refers to the way to how the air is obstructed; the second one gives us information about the places where this obstruction takes place.

2.2.1.2 The voicing mechanism

We can do another distinction between sounds depending on if the vocal cords vibrate or not during the sound production [14]. If these vibrate, then the sound is voiced. On the contrary, if the vocal folds do not vibrate, the sound is voiceless.

Voiced sounds, such as vowels and consonants like 'b' or 'd', have a quite regular time and frequency pattern, which is not like that in voiceless sounds. Moreover, the first ones typically have more energy than the second ones.

The vocal folds vibrate at rates from 60 Hz to 300 Hz. This rate of cycling (opening and closing) is what we call the *fundamental frequency* and will determine the higher frequency harmonics. It also contributes more than other factors to the perception of the *pitch*.

The harmonic of a wave is a component frequency of the signal that is an integer multiple of the fundamental frequency. For example, if the frequency is *f*, then the harmonics are at the frequencies *2f, 3f, 4f,* etc.

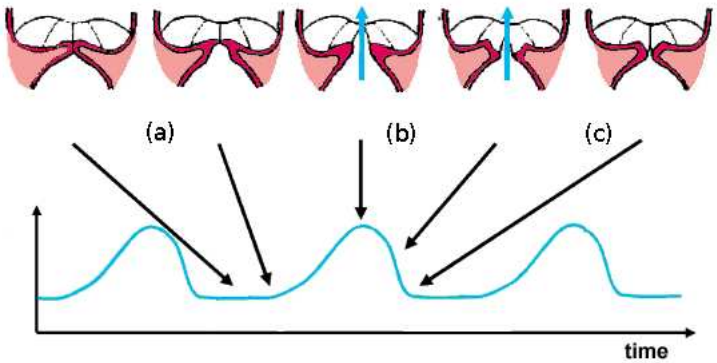

Figure 2.3: Air flow during vocal fold cycle

The vocal folds cycle is illustrated in Figure 2.4. At (a), the vocal folds are closed and the air pressure begins to increase until it overcomes their resistance. At this point, the folds are moved apart by the air blowing from the larynx. Then, at (b), the air goes out until the pressure decreases so much that the vocal folds fall back into the original position, (c), due to their natural elasticity. The period of the cycle depends on different factors, such us the amount of air pressure, that can be controlled by the speaker to vary the perceived frequency of a voiced sound.

2.2.1.3 Spectrograms and formants

The glottal wave [1] is periodic and consists of a fundamental frequency and a number of harmonics. Thus, and according to the Fourier theory, it can be analyzed as a sum of sine waves.

When we speak, the shape of the vocal tract changes and so do the resonances. Harmonics of the sound wave near the resonances are emphasized. The resonances of the oral cavities that depend on the configuration of the articulators are called *formants*. We can also define a formant as an energy concentration around a particular frequency in the speech wave [14].

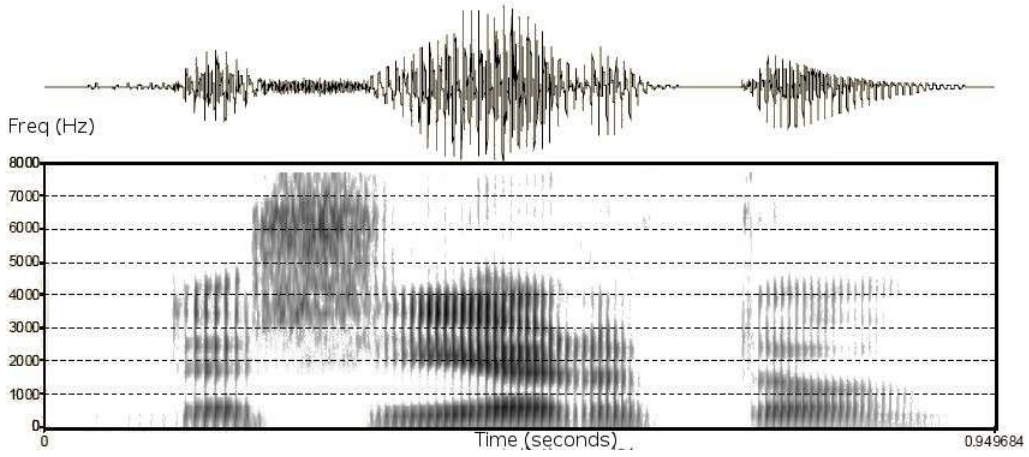

Figure 2.4: Spectrogram representation of a waveform

[1] the glottis is the space between the vocal cords

In the Figure 2.4, we can see a time waveform and its corresponding spectral analysis at each instant. A spectrogram of a time signal is a very powerful two-dimensional representation to see the spectral analysis of the time signal over the time. It displays time in its horizontal axis and frequency in its vertical axis. A gray scale is typically used to indicate the energy at each point (t, f), with white representing low energy and black high energy. Later, in Chapter 3, we will describe the mathematical methods to obtain the spectrogram of a time signal.

2.2.2 Speech perception

Speech perception refers to process by which humans are able to interpret and understand the sounds used in language.

The two main components of the auditory perception system are the peripheral auditory organs (ears) and the auditory nervous system (brain).

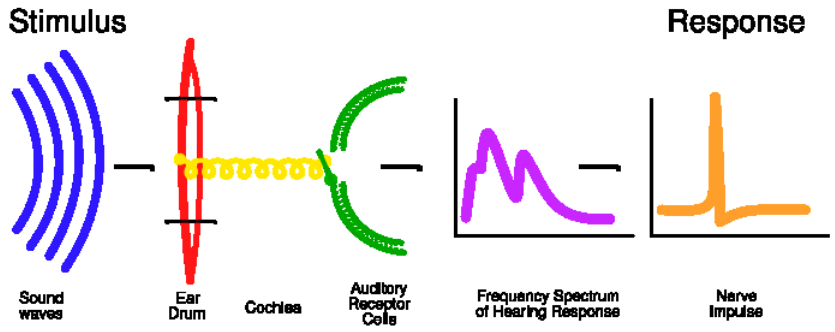

Figure 2.5: Representation of hearing

In the scheme of Figure 2.5, we can see the process by which an incoming acoustic pressure signal is transformed into a mechanical vibration pattern on the tympanic membrane[2] and then transmitted through the bones adjacent to it, called ossicles, until they reach the cochlea.

[2]colloquially known as the eardrum

The cochlea can be roughly regarded as a filter bank, which performs a spectral analysis of the incoming signal. Finally, this information is delivered to the auditory nerve to perform the perceptual extraction of the information.

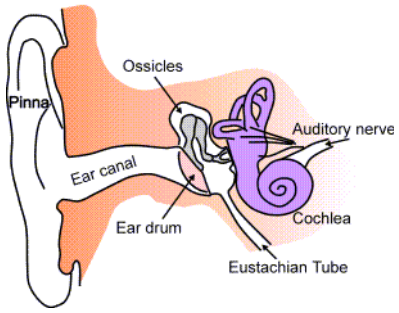

Figure 2.6: Anatomy of the human ear

The parts of the ear described above are shown in the graphic of Figure 2.6.

2.2.2.1 Physical vs perceptual attributes

Psycho-acoustics distincts between perceptual attributes of a sound and its measurable physical properties [10]. Each perceptual feature depends mainly on one physical attribute, but other properties may affect the perception. A list of such a relation is given below, in Table 2.2.

Physical quantity	**Perceptual quality**
Intensity	Loudness
Fundamental frequency	Pitch
Spectral shape	Timbre
Onset/offset time	Timing
Phase differences	Location

Table 2.2: Relation between perceptual an physical attributes of sound

One difference between physical and perceptual attributes is the phenomenon known as the non-uniform *equal loudness* perception of tones of differing frequencies. The reason is that the sensitivity of the ear varies with frequency . That is, two sounds with the same intensity are not perceived equal if their frequencies are different. The graph of equal loudness curves is shown in Figure 2.7.

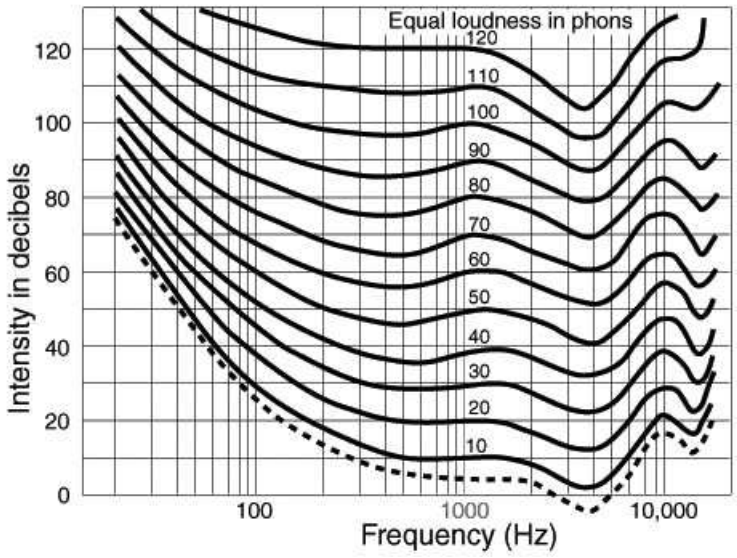

Figure 2.7: Equal-loudness curves

Another interesting fact is the human ability to distinguish two sounds, with the same loudness and pitch, produced by different objects. It is exactly the same with speech. The same message produced by different people sounds different. The perceptual attribute behind it all is the *timbre*.

2.2.2.2 Frequency analysis

The auditory system acts as a spectral analyzer of sounds. For that reason researchers attempt to find frequency scales that model the response of the human perceptual system.

The concept of critical band is very important to understand auditory phenomena such as loudness, pitch and timbre.

The cochlea can be seen like a bank of overlapping filters with bandwidths equal to the critical bandwidth. The values in the table below correspond to the *Bark frequency scale* (BFS).

Bark band	**Edge (Hz)**	**Center (Hz)**
1	100	50
2	200	150
3	300	250
4	400	350
5	510	450
6	630	570
7	770	700
8	920	840
9	1080	1000
10	1270	1170
11	1480	1370
12	1720	1600
13	2000	1850
14	2320	2150
15	2700	2500
16	3150	2900
17	3700	3400
18	4400	4000
19	5300	4800
20	6400	5800
21	7700	7000
22	9500	8500
23	12000	10500
24	15500	13500

Table 2.4: Critical bands. The Bark frequency scale

The objective in using this scale is to achieve similar results to those that ear does when it processes the spectral information.

Another perceptually motivated frequency scale is the Mel frequency scale (MFS) [18], which models the non-linear frequency resolution of the human ear. It is linear up to 1 kHz and logarithmic above. As the rest, the Mel scale attempts to model the sensitivity of the human ear better than the linear scales. It has been used a lot in modern speech recognition systems.

2.2.2.3 Masking

Sometimes, a clear and audible sound is masked by other louder sound. When this two sounds occur simultaneously, this phenomenon is called simultaneous or frequency masking.

Then, we can say that the frequency masking takes place when one sound cannot be perceived if another sound, intense enough, is in the same critical band.

Chapter 3

Acoustic feature extraction

One of the major requirements in the speech/non-speech discrimination problem is to construct a data set, from the audio input, in such a way that simplifies the task of the classifier algorithm. To this end, the psychoacoustically motivated amplitude modulation spectrogram (AMS) is used. Actually, it is the most common technique in current Automatic Speech Recognition Systems (ASR).

The following lines try to give a general view of the modulation spectrum. First of all, a description of the relevant components of the speech spectrum is given. After that, we review the basic mathematical concepts behind the method and finally, some hypothesis, drawn from the observation of diverse acoustic situations, are presented.

3.1 The modulation spectrum

3.1.1 Introduction

Our hearing system is quite sensitive to changes of signal energy over time, and a lot of information of speech is encoded in them. As those changes are much slower than the frequencies that carry acoustic signals, they are termed modulations, or amplitude modulations. They can be different in the different carrier frequency bands.

Studies on human speech perception show the importance of the previously mentioned slow changes in the speech spectrum. This is due to the rate of change of the vocal tract shape. Such changes correspond to low-frequency

amplitude modulations with rates below 16 Hz.

Traditional representations of speech tend to emphasize the minute spectro-temporal details of the speech signal. On the contrary, the modulation spectrogram [8] is a kind of representation, which tries to be insensitive to the speaker variability and acoustic distorsion. A key step to do that is to focus on the elements of the signal encoding phonetic information and suppress the irrelevant ones.

With this, the modulation spectrum of continuous speech can be seen as a way of displaying and encoding the signal in terms of the distribution of slow modulations across time and frequency [8]. It can be obtained by the spectral analysis of temporal trajectories of spectral envelopes of speech [9]. It is mainly constituted by components between 2 Hz and 16 Hz, reflecting the syllabic and phonetic temporal structure of speech [7]. It is interesting to indicate that the human auditory system is most sensitive to modulation frequencies around 4 Hz [4] [11] and that significant speech intelligibility remains even if only modulations up to 16 Hz are preserved [1].

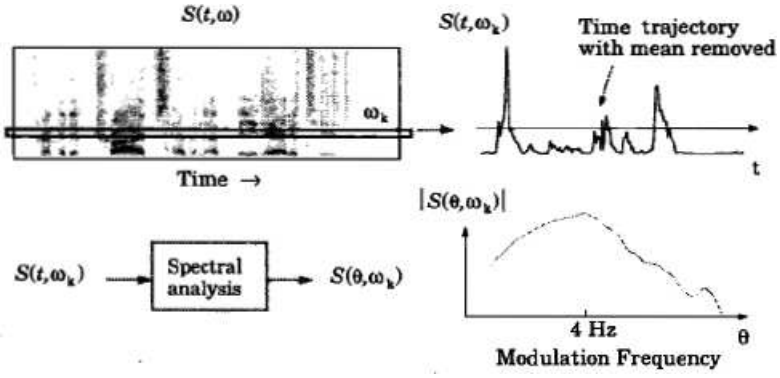

Figure 3.1: The modulation spectrum of speech [9]

The figure above describes the different steps that allow to go from the time representation of the signal to the modulation spectrum of speech. The process can also be regarded as the filtering of each time sequence of the spectral parameters (TSSP). For that reason, the TSSP spectrum has been called modulation spectrum [16].

3.1.2 Algorithm

The algorithm used to compute the modulation spectrum is derived directly from the definition given above and from [2]. This section gives general definition, whereas the concret parameters are described in detail in 5.2.2:

Given a digitalised audio signal, $x[n]$, the MS is obtained by:

- Compute the spectrogram of the signal, following the next steps:

  1. Break the signal into overlapping chunks, $x_m[n]$, where m is the chunks index.
  2. Calculate the short-time Fourier transform ($STFT$) of each chunk:

$$STFT\{x_m[n]\} \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x_m[n]\, e^{-j\omega n}$$

  If we define the short-time signal $x_m[n]$ as

$$x_m[n] \equiv x[n]\, w_m[n]$$

  the product of $x[n]$ by a *Hamming window* function $w_m[n]$, which is same for all chunks (1) and equals zero everywhere except in a few samples (2):

$$(1)\ w_m[n] \equiv w[m-n]$$
$$(2)\ w[n] = 0 \ \ \forall\ \ |n| > N/2$$

  Then:

$$x_m[n] = x[n]\, w[m-n]$$

  Thus, we can finally define the $STFT$ as:

$$X(m,\omega) = \sum x[n]\, w[m-n]\, e^{-j\omega n}$$

  And the magnitude of the spectrogram at each point (t, f)correspond to the magnitude of the $STFT$ of the corresponding chunk:

$$s\{x_m[n]\} \equiv |X(m,\omega)|^2$$

- Compute the modulation spectrum:

  For $k = 1, \ldots, the\ number\ of\ frequencies$:

  1. Compute Fourier transform at the present center frequency, ω_k, along the time dimension of the spectrogram:

$$MS\left\{x\left[n\right]\right\} \equiv X\left(\theta, \omega_k\right) = \sum_{m=-\infty}^{\infty} \left|X\left(m, \omega_k\right)\right|^2 e^{-j\theta m}$$

  2. Compute modulation power over center frequency, modulation frequency and time:

$$\left|X\left(\theta, \omega_k\right)\right|^2$$

3.1.3 Experiments

Once we have described the main features of the MS and why we use it, we would like to see what we obtain from it. That is, given some acoustic events, how does the MS look like?

To that end, we choose the audio files to be enough representative of all the situations that we want to be able to detect. Single and several persons speaking, one or multiple cars moving in presence of speech or not, and environmental noise are the selected situations.

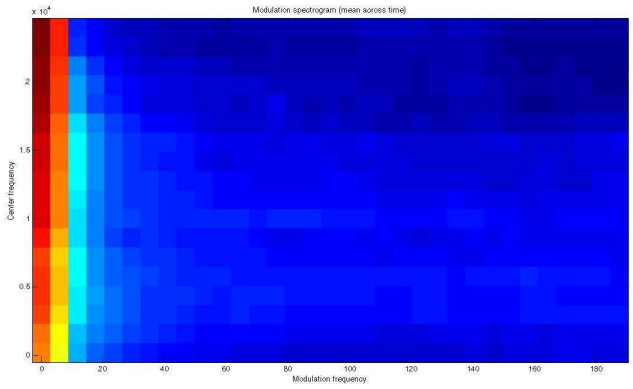

Figure 3.2: Modulation spectrogram of a single person speaking

The Figure 3.2 shows the MS of a single person's voice recorded in an isolated environment. The Figure 3.3, in contrast, shows a conversation that takes place between several people.

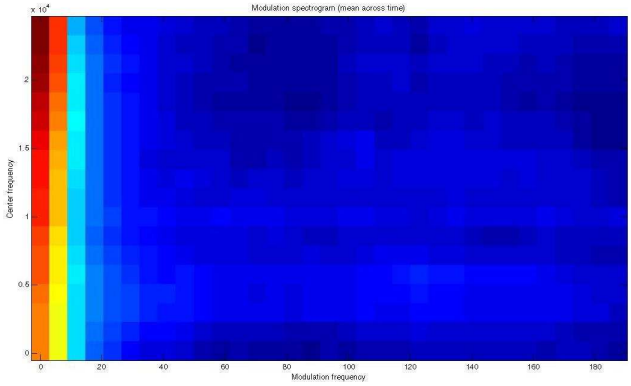

Figure 3.3: Modulation spectrogram of several persons speaking

The next two figures represent the MS of single and multiple cars in movement. In comparison with the previous situations, now, we can't observe the presence of an energy bar (light blue) at modulation rates round 10 Hz.

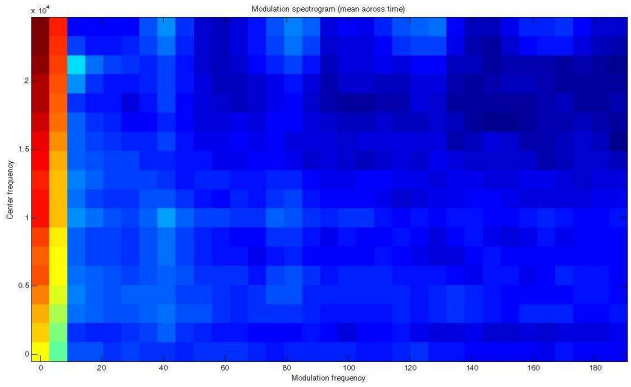

Figure 3.4: Modulation spectrogram of a car moving

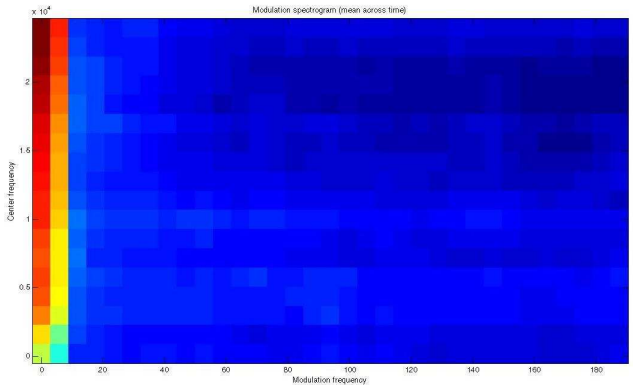

Figure 3.5: Modulation spectrogram of several cars moving

In the following situation, a man speaking is recorded while several cars are moving. This time, the MS contains again an energy bar at the mentioned frequencies in the first experiments.

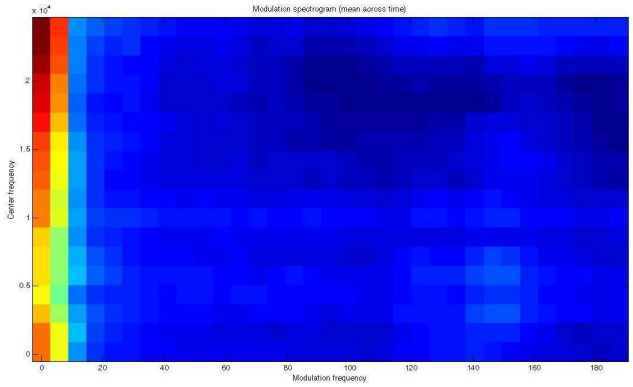

Figure 3.6: Modulation spectrogram of a single person speaking and several cars moving

Finally, with environmental noise, the MS graphics are very similar to those produced by cars. No energy bars at rates of 15 Hz are present.

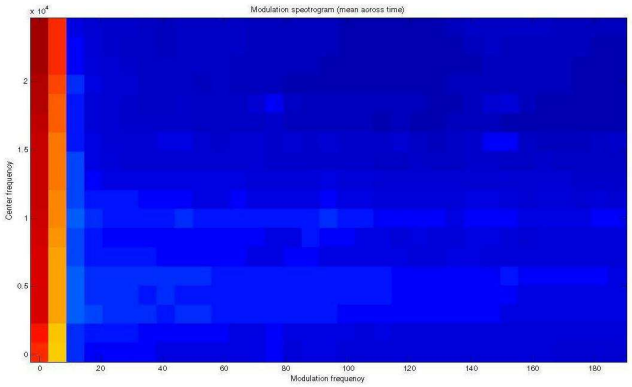

Figure 3.7: Modulation spectrogram of environmental noise

3.2 Conclusions

There is a common feature observed for all the experiments in which a single person o more persons are speaking. For all the carrier frequency bands, and for modulation frequencies between 10 - 15 Hz, the energy is lower than that for frequency rates below (light blue energy bar), but higher than the energy of the modulation frequencies above 15 Hz. On the other hand, this fact does not occur in non-speech sounds.

Therefore, it seems that the presence of energy in this range of modulation frequencies discriminates quite well speech sounds. For that reason, it is expected that the best selected features by the classifier algorithms are going to be in that range.

Chapter 4

Pattern recognition

The sound processing task relies heavily on pattern recognition, which is in fact, one of the most challenging problems for machines. We would really like to incorporate the ability to reliably recognize patterns or regularities in the nature to our work and life, so machines become much powerful and easier to use.

There are many learning algorithms and techniques for pattern recognition and one normal question that can be formulated is which one is "best". If one studies in depth the problem, will see that no pattern recognition method is inherently superior to any other. It is the nature of the problem, prior data distribution and other information, the ones that determine which algorithm is going to provide the best performance. In spite of that, some algorithms can be chosen because of their lower computational complexity; others because they take into account some prior knowledge of the form of the data.

In this chapter, we will concentrate in one technique called AdaBoost [6]. Such technique is an adaptative boosting algorithm. It is interesting because of two main reasons. In first place, it focuses on the most difficult examples of available data, so that they can be classified. And in second place, it does not depend upon a particular classifier or learning method. For that reason, it can be used with different algorithms. Then, in the last part of the chapter, we will discuss a way to validate the analysis through the *cross-validation (CV)* method. Later, in the next chapter we will compare this technique to others used in similar works, e.g. the *support vectors machine* (SVM), to see if the results provided by the first one are similar to those achieved by the second one.

4.1 Introduction

4.1.1 Basic concepts

First of all, we shall review some basic concepts about classifiers and pattern recognition in general [5].

A pattern is a set of instances which share some regularities and are similar to each other. A pattern usually happens repeatedly. A pattern can also be seen as a feature present in the set of elements or classes that we want to be able to recognize. We will work with data bases and vectors. Each dimension of the vector corresponds to a feature.

Mathematically, we can define the previous concepts as it follows:

1. Patterns: $x_i \in X = R^d$
2. Classes: $y_i \in Y = \{-1, +1\}$, because we are going to work with 2 classes: speech and non-speech.
3. Training data set:$(x_1, y_1), \ldots, (x_N, y_N)$
4. Learned function: $f : R^d \rightarrow R$
5. Classifier: $h_f(x) = sign(f(x)) : R^d \rightarrow Y$

It would be desirable to have a great number of vectors, each one of which with so few dimensions as it is possible. Thus, the task of the classifier is easier in the sense that, given a fixed number of vectors or points, it is simpler to handle them when the dimension of the space in which we are working is low.

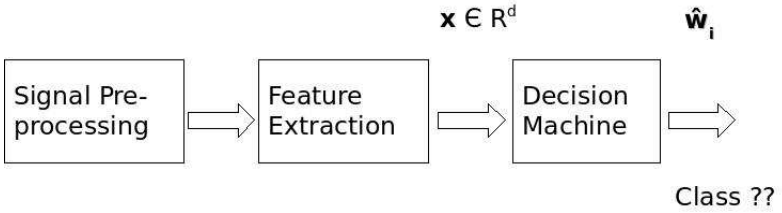

Figure 4.1: The classification process

The figure above shows the classification process with its corresponding steps. First of all, a pre-processing is required in order to prepare the signal. We need to convert the physical inputs into signal data. In our case, the sounds recorded with a microphone are transformed into digital signals, after individual processes of sampling and quantification. After that, the feature extractor measures object properties that can be useful for the classification process. The modulation spectrum, described in detail in Chapter 3, performs such a task and provides a set of vectors to be analyzed by the AdaBoost classifier. This one uses these features to assign the vectors obtained before to a category, that can be speech or non-speech.

4.1.2 Feature selection

Sometimes, the design of the feature extractor can have more influence on the classification error than the classifier itself.

A low number of features results in:

- Simpler decision boundaries
- Simple structure of classifiers

It is interesting to find distinguishing features to be able to perform the classification with the minimum number of them. Features must provide similar measured values for objects pertaining to the same category and different values between objects of different categories. That is, high variability inter-classes and low variability intra-classes.

4.1.3 The classifier

The task of the classifier is to use the feature vector provided by the feature extractor to assign the object to a category.

Part of the difficulty in the classification task depends on the variability in the feature values for objects in the same or different category. This variability is due to noise, which can be classified to the intrinsic noise to the classes or the noise due to the erroneous measurements.

The next diagram shows the traditional steps in a classification process. First of all, the data base must be divided into a set of training and a set of test. After that, we choose and train an algorithm on the training set. Finally, we evaluate the classifier on the test set.

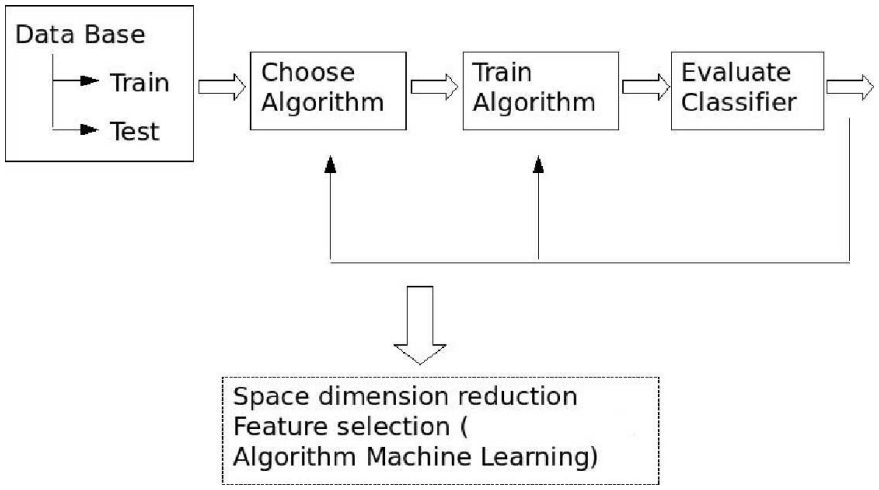

Figure 4.2: Steps to design the classifier

The first step in the process is to split the available data into training and test sets. The first group will be used to set the classifying function and the decision boundaries. These must be suitable for our data. The second set will be used to verify the previous results.

After we choose an algorithm, we have to train it. In this phase, a series of parameters must be adjusted for the determined application from a training set by means of feature vectors. As we are only going to work with supervised learning, we know a priori the class or the category to which such vectors belong to. That means that we have a vector of labels, $y_i \in Y = \{-1, +1\}$, with information on the class for each vector.

The evaluation of the classifier must be done according to the selected design criterion: either the minimun classification error or the minimum of a cost function.

After all this process, we should obtain something similar to what we can see in the illustrative example of Figure 4.3. In this situation, we assign one of the three possible categories to each one of the vectors in the data set. On the contrary, in our work, although the proceeding is the same, we work with 2 classes instead of working with 3. These classes are speech and non-speech and to detect them correctly constitutes the core of the current work.

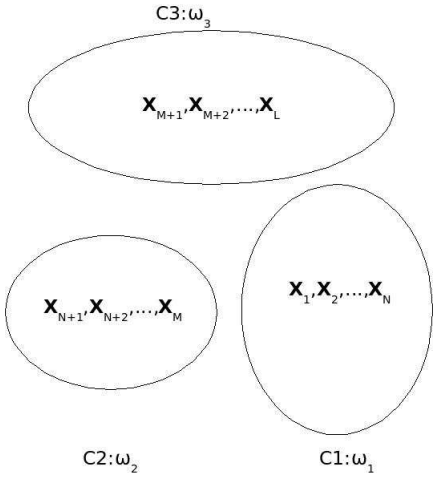

Figure 4.3: Process of assigning a category, 3 in this example, to each one of the vectors of the set

4.2 Boosting

Boosting is a machine learning meta-algorithm, developed to improve the accuracy of any given learning algorithm. It is based on the fact that it is possible to find a better classifier by combining "weak" classifiers [3].

A weak classifier is a classifier usually chosen to be simple and often works better than a random classification.

4.2.1 Boosting algorithms

Most boosting algorithms are iterative and add weak learners at each round to build a final strong learner. At every iteration, a weak classifier learns the training data with respect to a distribution and then is added to the final strong classifier. A way of doing that is by means of weighting the weak learners and the data points, using the weak learner's classification error. This means that the misclassified vectors gain weight and the correctly classified lose weight. Thus, the successive weak classifiers will focus more on the previously misclassified vectors to try to correctly assign them to their corresponding category in the next rounds. After adding a new weak learner

to the final strong classifier, the data weights are changed in order to focus on the still misclassified examples.

There are many boosting algorithms. The original ones were proposed by Rob Schapire and Yoav Freund and were not adaptative. The main difference between them is their way of weighting training data points. AdaBoost, LPBoost and TotalBoost are the most popular.

The following lines will concentrate on the AdaBoost algorithm [6].

4.2.2 AdaBoost

4.2.2.1 General description

Adaboost means Adaptative Boosting and was formulated by Yoav Freund and Robert Schapire. It is an algorithm that can be used with other learning algorithms to improve their performance.

AdaBoost is a way for constructing a strong classifier as a linear combination of weak classifiers:

$$f(x) = \sum \alpha h_t(x)$$

where $h_t(x) : X \rightarrow \{-1, +1\}$ is the simple classifier. An example can be seen in Figure 4.4, where a 1-d weak learner separates the points from 2 classes.

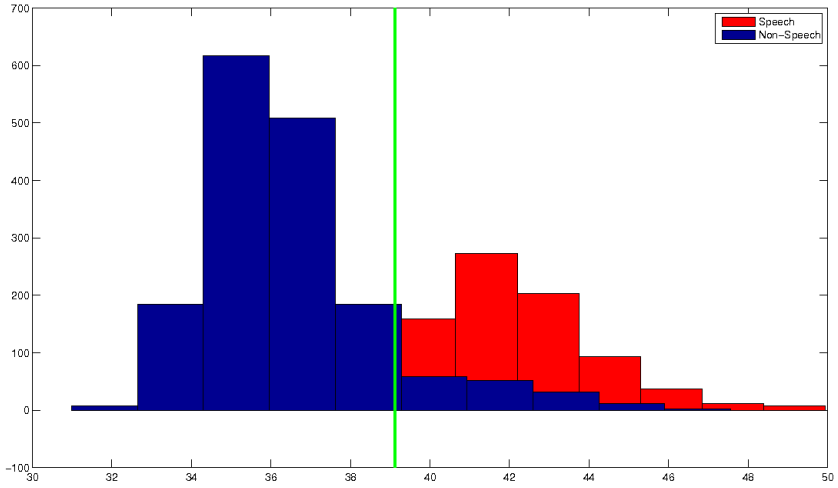

Figure 4.4: Weak classifier separating two classes

AdaBoost is adaptative in the sense that subsequent classifiers are built with more emphasis on the misclassified data. In this way, it can be said that it focuses on the informative or difficult patterns. Furthermore, so many weak classifiers as it is necessary can be added until the desire classification error is achieved.

4.2.2.2 Algorithm [5]

Given: $(x_1, y_1), \ldots, (x_m, y_m)$, where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize the data weights $D_1(i) = \frac{1}{N}$
For $t = 1, \ldots, T$:

- Find the classifier $h_t(x)$ that minimizes the error with respect to the weights D_t:

$$h_t = \arg\min_{h_j \in H} \varepsilon_j = \sum_{i=1}^{m} D_t(i) \left[y_i \neq h_j(x_i) \right]$$

- Continue if $\varepsilon_t \leq 1/2$, otherwise stop
- Choose $\alpha_t \in \mathbb{R}$, typically $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$, where ε_t is the weigthed error rate of classifier h_t
- Update the data weights:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & h_t(x_i) = y_i \quad (if\ correctly\ classified) \\ e^{\alpha_t}, & h_t(x_i) \neq y_i \quad (if\ incorrectly\ classified) \end{cases}$$

where $Z_t$is a normalization factor.
Finally, the final output classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

4.2.2.3 Method description

We start with a data distribution that belongs to the training set. To make things clearer we will use an illustrative example of a 2-d space with vectors of two classes (in our situation, it would be speech/non-speech).

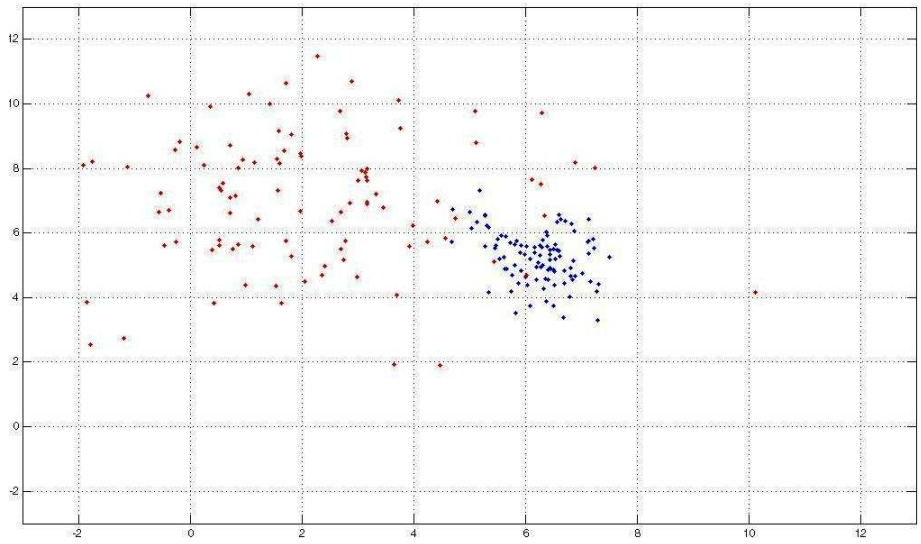

Figure 4.5: Illustrative example of a data distribution in a 2-d space, to be classified by the AdaBoost algorithm

The figure above shows two classes distributed randomly. In principle, it is not obvious to see how a linear classifier could separate the two classes.

We initialize the data weights $D_1(i) = \frac{1}{N}$, so that all the points in the space are equally important.

We iterate for all the classifiers, For $t = 1, \ldots, T$:

1. Train a weak learner using the data weights, D_t, and obtain h_t.

   (a) At the beginning, with T = 1, all the vectors or examples in the space are equally probable.

   (b) In the next rounds, it is more probable to select the misclassified examples (those which make the classifier fail).

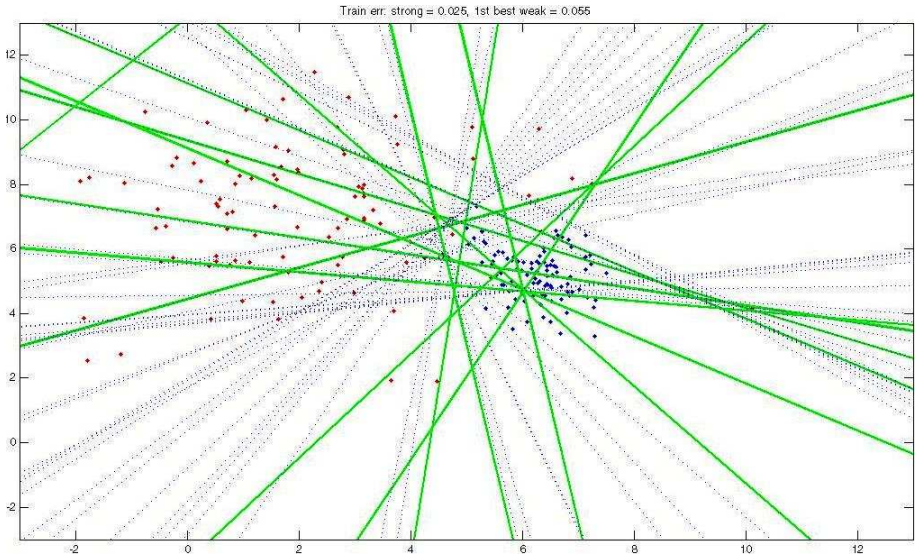

Figure 4.6: Best T weak classifiers selected after the training phase in the AdaBoost algorithm

2. Choose a weight (confidence value) $\alpha_t \in \mathbb{R}$. It is the relative importance of the weak classifier in the round t. Those classifiers that achieve lower classification errors are weighted higher.

   (a) If ε_t is the error associated to h_t

   $$\varepsilon_t = PrD_t \left[h_t \left(x_i \right) \neq y_i \right]$$

   (b) The value of α_t comes from trying to optimize such error and equals to:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right) > 0$$

3. Update distribution over training set. We will increase the weigths of the misclassified examples by the current weak classifier,h_t, and decrease the weights of the correctly classified examples.

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} A$$

where A can take the following values:

$$if\ h_t(x_i) = y_i \Longrightarrow A = e^{-\alpha_t}$$

$$if\ h_t(x_i) \neq y_i \Longrightarrow A = e^{\alpha_t}$$

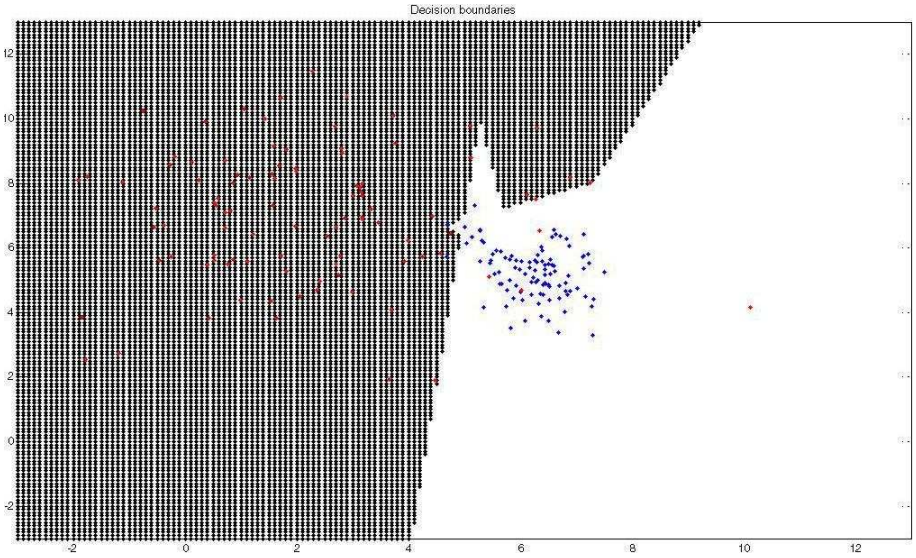

Figure 4.7: Decision boundaries given by the strong classifier in the AdaBoost algorithm

In Figure 4.6, we can see the decision boundaries originated by the strong classifier. It is possible to appreciate that the separation of the classes is better performed than with one single weak learner.

Finally, the classification error can be calculated by counting the differences the number of misclassified examples.

4.3 Data validation

We are interested in finding the generalization rate of a classifier on a given problem to see if it performs well enough to be useful.

Data validation is the process of ensuring that a program, routine or algorithm works with clean and useful data. In other words, it verifies that the data used as an input for the system is correct and meaningful. Although there are different types of routines, we are only going to work with the so-called *cross-validation* technique.

We randomly split the initial sample of data into two subsets: one is used as the training set for adjusting learning model parameters in the classifier. The other set is the *validation* or *testing set* and it is used to confirm and validate the initial analysis and to estimate the generalization error. The algorithm is trained until we reach the minimum validation error.

The *K-fold cross-validation* is a generalization of the method described above in which the original data set is partitioned into K disjoint subsets of equal size N/K, being N the total number of examples. Of the K subsets we use K-1 to train the model an the other one to test it. The cross-validation process is then repeated K times (*folds*), with a different validation set, of the total K subsets, each time. The K results from each iteration can be averaged or combined to produce a single estimation.

Part II

EXPERIMENTAL WORK

Chapter 5

Experimental evaluation

In this section, we use the background theory described in previous chapters to build a system able to discriminate speech from other sounds. The purpose of the experiments is to see the results achieved by the AB technique and to compare them with those obtained by related works [17].

We start describing the data used in the classification process: the kind of experiments, the way to collect it and how we use it. Then, we give a brief description of the method used in this work, as well as the parameters chosen for the signal processing techniques and the classification algorithms. Finally, we present and discuss the experimental results.

5.1 Data acquisition

5.1.1 Introduction

We have worked with two different sets of audio signals. The first set, or the preliminary one, was used to observe the different kind of signals on the frequency domain and derive their main properties. With the help of the modulation spectrogram, we could make some hypothesis (see 3.1.3). For modulation frequencies round 10 Hz, we observed that speech signals presented a higher power level than the rest of sounds. Thus, it wouldn't be surprising to find distinguishing features at that frequency rates. This fact will be confirmed in further sections. The second set was used to train and adjust the parameters of the AB classifier and also to test or validate the results.

All the experiments were performed by means of a digital camera Canon XM 1 at a sampling rate of 48 kHz. We recorded both, audio and video, separated them into different files and concentrated solely in the audio ones.

Figure 5.1: Digital camera Canon XM 1

5.1.2 Preliminary data set

The experiments consisted in different recordings, about 30 seconds each one that, later on, would allow us to see the spectral properties of the sounds that we want to be able to recognize at the end of this work. These experiments are the following:

- Single person speaking.
- Single person speaking and different background tones.
- Several persons speaking.
- Several cars.
- Several cars and a single person speaking.

Such experiments were recorded in three places. Single and several persons speaking, in isolated and coffee rooms respectively; Experiments related to cars, in one concurred street in the Prague city, with continuous traffic.

5.1.3 Training and test set

For the AB algorithm training and test phases, a bigger amount of data is required. Now, we will use 45 minutes of audio divided as it follows:

- 15 minutes of several persons speaking.
- 15 minutes of several cars
- 15 minutes of environmental noise

The first experiment was carried out in a coffee room, the second one, as in 5.1.2, in one concurred street in Prague, and the third one, in a quiet place with none of the previous kind of sounds.

5.2 Method description

5.2.1 General principles

The method used for the speech/non-speech discrimination task is based on the following general ideas:

- Spectral analysis of the incoming audio files.
- Mel-scale transformation.
- Computation of an average modulation spectrum for speech and non-speech sounds.
- AdaBoost decision algorithm.

5.2.2 Method used in the present work

The modulation spectrum of the incoming audio data is obtained by the spectral analysis of temporal trajectories of spectral envelopes of speech. It is accomplished by means of two FFT-calculations in the following way:

1. The incoming audio, sampled at 48 kHz and properly labeled, is loaded into a vector. Then, it is divided into chunks of one second length. In order to minimise the edge effect, each one of these chunks overlaps

with the previous one a 50 % of the time. Such chunks will be assigned to the same class as the sound that they come from. They are the basis for the speech/non-speech decision task.

2. Then, the short-time Fourier transform (STFT) or spectrogram is computed for each chunk. It uses a Hamming window to compute the FFT over 256 points. Again, the overlapping between adjacent frames is used to reduce the edge effect, by shifting windows 128 samples or the 50 %.

3. The Mel-scale transformation is applied to the magnitude vectors resulting from the previous analysis. It converts the spectrogram, with linear frequency axis, to one with logarithmic "Mel" frequency axis, to approximate the frequency response of the ear, linear up to 1000 Hz and logarithmic thereafter.

4. After that, the modulation spectrum is performed by computing the spectrogram over all input signal's subbands. A Hamming window, shifted by 48 samples (75 %), is used to compute the FFT over 64 samples.

5. For each incoming audio signal, this process returns a 18*33 dimensions matrix that can be seen as a vector with 594 features. This vector consists of the modulation power over each center frequency.

6. Since the dimension is very big and to make the classification problem more accessible, we will reduce the number of features in the following way:

   (a) Keep the modulation frequencies up to 40 Hz, because the useful information is contained in them. This means that we will use the first 8 frequencies of the 33 returned by the previous analysis.

   (b) Make an average of the central frequencies in such a way that one from each pair of consecutive frequencies is obtained. Thus, we are going to transform the initial 18 ones into 9.

   With these two steps we have achieved 72 feature vectors and, although they are still big, they are much smaller and manageable than the original ones.

7. The total number of vectors, which depends on the data used in each experiment, will be randomly divided into two subsets: training and test.

Once we have completed the feature extraction, it's the turn for the feature selection and the classification. That is, finding the minimum number of features that allow us to achieve a fixed error. To this end, we will use the AdaBoost technique (see 4.2.2), which builds a strong classifier as a linear combination of weak or simple classifiers.

The first step is to construct the set of weak learners. Our starting point will be 72 dimensional planes, parallel to the axis of the features. Since the present work tries to be a reference for future works, we will start using very simple weak classifiers. In particular, we will use projections of the planes over all and each one of the 72 dimensions. The result will be one random line by each dimension.

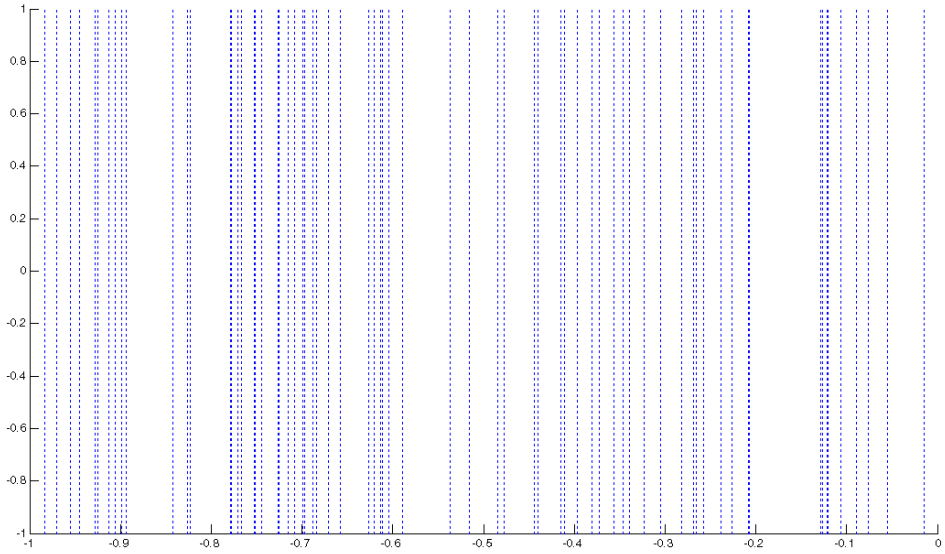

Figure 5.2: Initial 1-D random thresholds in the classification algorithm

The Figure 5.2 shows the mentioned thresholds plotted all together on the same graphic. It does not mean that these are projections over the same coordinate, but a means to visualize them simultaneously.

After we have got the random thresholds, they must be trained in such a way so that, for each dimension, the points from the two possible classes are separated in an optimal way. In other words, given a projection of the data set over one dimension, the AB algorithm trains a weak classifier (single threshold) and places it where fewer misclassified examples are obtained.

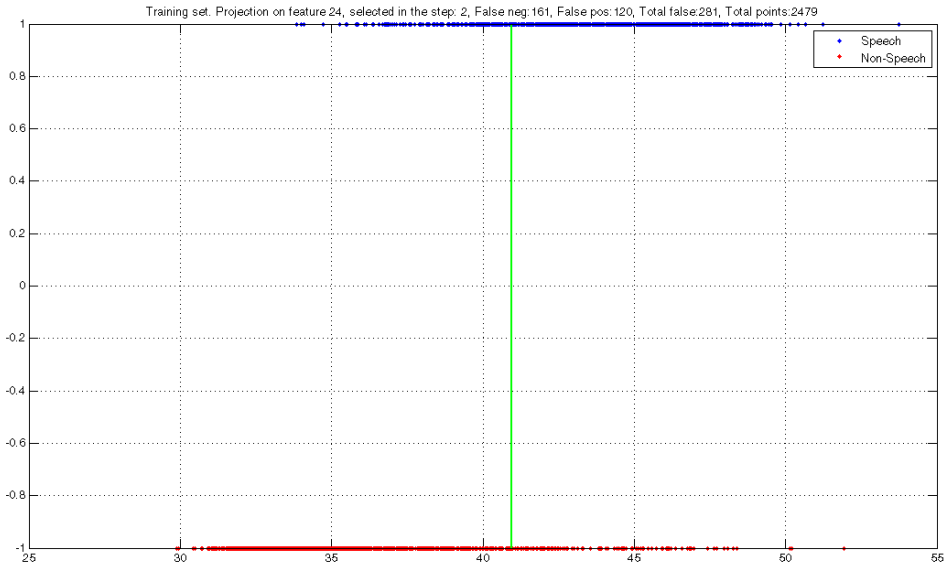

Figure 5.3: Data projection over 1-D and misclassified points with one weak learner

The figure above shows the points corresponding to the vectors of the training set, projected over one of their dimensions. False negatives are the speech examples classified as non-speech; false positives are the non-speech examples classified as speech. The weak classifier is trained until the lowest classification error is achieved.

The Figure 5.4 shows the best weak classifiers for a given training set. These will be used by the algorithm to build the strong classifier, which will be used for the speech/non-speech discrimination task. Again, they are plotted together on the same graphic.

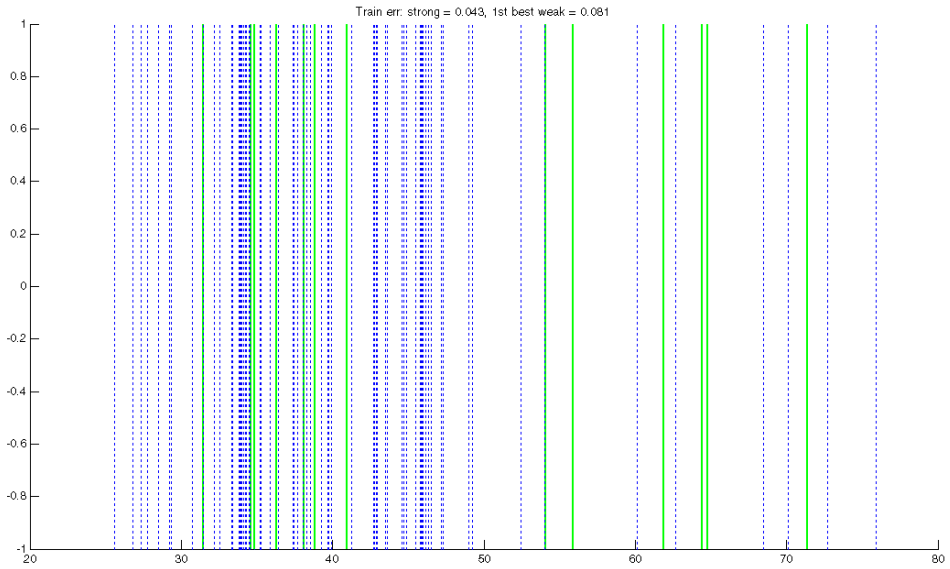

Figure 5.4: Best weak classifiers selected by AB to build the strong classifier

Finally, the total training and test errors come from the difference between the labels assigned by the classifier to each chunk or vector of the two data sets, and the original labels, manually entered before the process.

5.3 Experimental results

5.3.1 Classification errors

At this point, we have already described the entire process. We have seen how the data was collected and prepared to extract features by means of the modulation spectrogram. The classification method, with the AdaBoost algorithm, has been described too.

Next we give the description of the different tests carried out and their performance. To make it clearer, ROC curves, training and test error graphics and the best selected features, are also presented.

For all the experiments, we will use the training and test data set, described in 5.1.3. That implies that, if we use one second length chunks,

overlapped a 50 %, we will get 5145 vectors with 72 dimesions. These vectors will be randomly divided into two halfs. One half will be assigned to the training set and the other one to the test set.

In the first experiment, we will perform 20 analysis, with both training and test phases. For each analysis, a different split of the data set, with different training and test sets each time, will be used. Thus, we will be able to observe a more general behaviour of the algorithm.

The Figure 5.5 represents the ROC curves obtained from 20 analysis, using a strong classifier formed by 40 weak learners. It represents false negatives versus false positives. False negatives stand for examples labeled as speech but classified as non-speech. On the contrary, false positives stand for examples labeled as non-speech but classified as speech. Finally, the blue points represent the train errors and the red ones, the test errors.

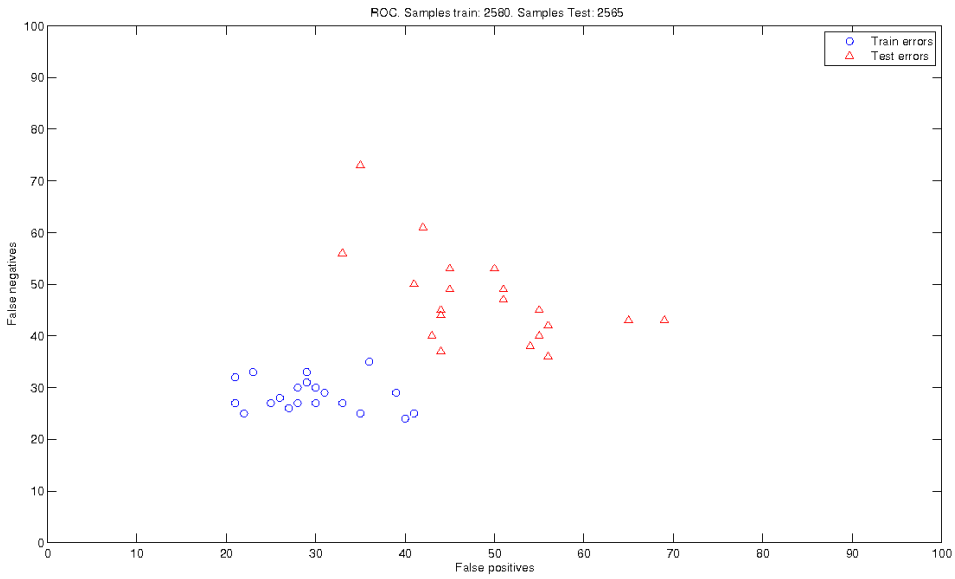

Figure 5.5: ROC curves for 20 analysis performed with a strong classifier formed by 40 weak classifiers

The first that can be said about the graphic is that test errors are greater than train errors. Furthermore, once the points that look anomalous have

been discarded, we can observe that test errors are much more dispersed than the training ones. This is all coherent, since the AB algorithm focuses only on the training set in order to adjust the parameters of the classifier and build the decision boundaries. For that reason, the points of this group are better classified than the rest.

Finally, we can see that the errors achieved using 40 features are relatively low: 2,5 % - 3 % for the training set and 3,5 % - 4 % for the test set.

In addition, we performed another experiment consisting in 100 analysis and using a classifier formed by 40 weak learners as well. The results are similar to those in the previous experiment.

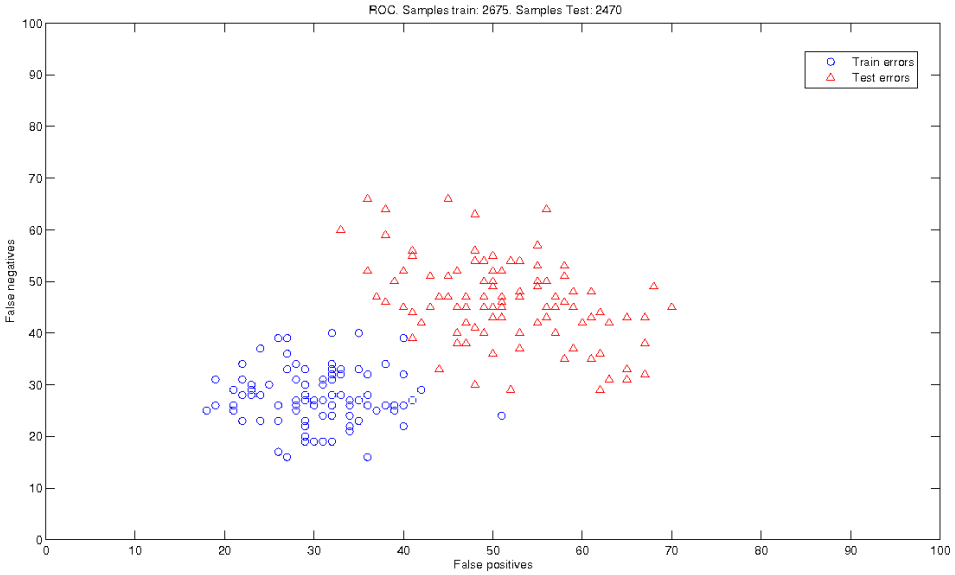

Figure 5.6: ROC curves for 100 analysis performed with a strong classifier formed by 40 weak classifiers

The following picture shows the curves for the training and test errors as a function of the number of features selected. Each one represents a different data split. Again, the blue and red lines represent the training and test error, respectively.

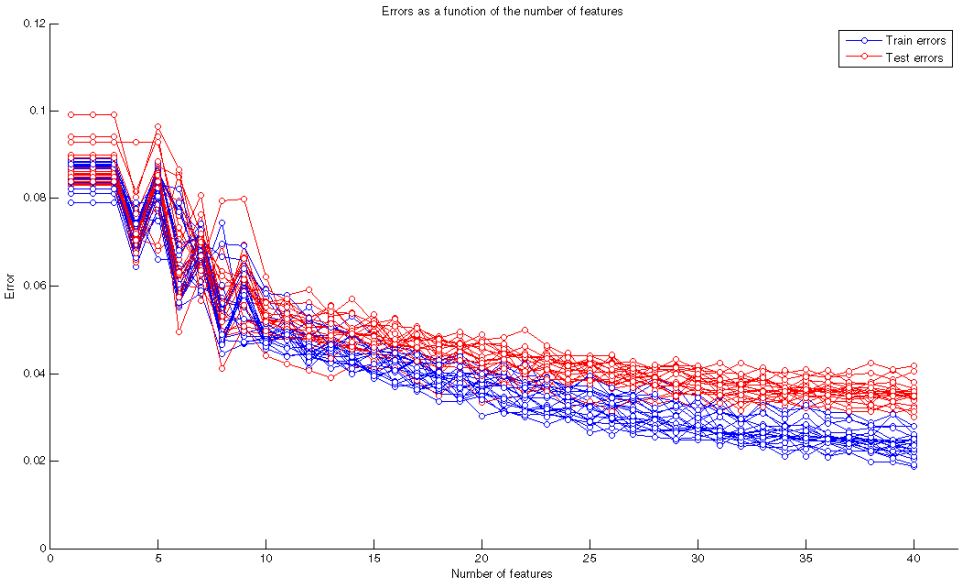

Figure 5.7: Training and test errors curves, for 20 analysis, as a function of the number of weak classifiers selected to build the strong one

It is very difficult to draw some conclusion from this graph because there are many curves and the errors cannot be seen clearly. For that reason, an statistical analysis will be performed over the results by means of the matlab tool, *boxplot*. At every point of the horizontal axis, a box with a line inside is plotted. It represents the median of the train or test errors and its dispersion around the median value. Outliers are values beyond the box limits and probably represent anomalous situations that must be discarded. The statistical analysis can be seen in Figure 5.7 and in Figure 5.8, for the train and test errors, respectively.

The Figure 5.7 shows the boxplot analysis of the training errors achieved by the classifier as its number of components increases. We can distinguish three different regions. The first one, from one to three features, is a constant section. Up to 9 features, we can observe an oscillating error. From this point, the error shows a decreasing tendency, that is somehow similar to an inverse exponential function. It is significantly decreasing up to 27 features. From here on, the increase of the number of features entails a very small diminution

of the error. It is only a 0,6% when all 40 features are used. It is not much if we think in the computational complexity added to the problem.

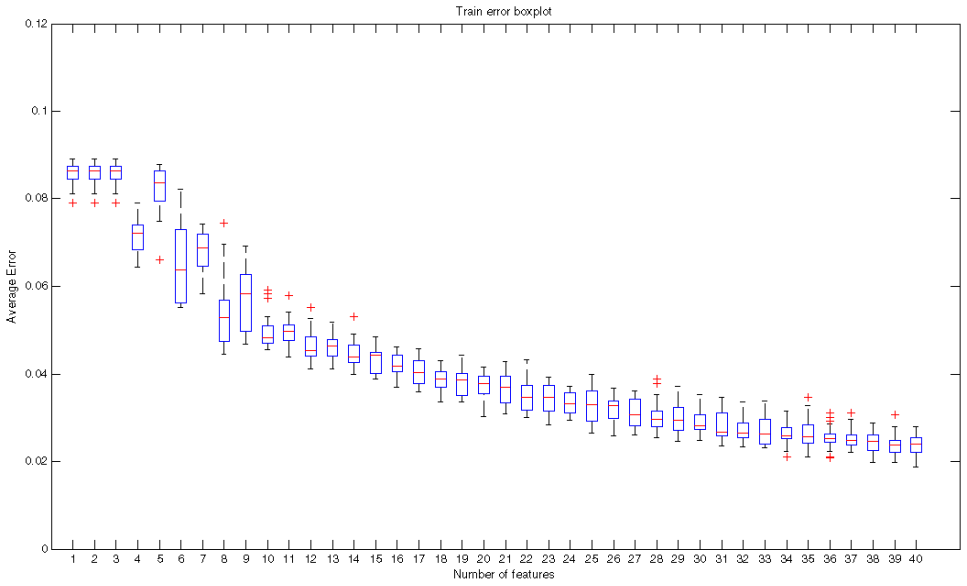

Figure 5.8: Statistical distribution of the training error curves as a function of the number of weak classifiers selected

Finally, the test error boxplot is shown above. This situation is quite similar to the one described in Figure 5.7. There are also three different regions. The first one is constant up to 3 features. The second one presents an oscillating error up to 9 or 11 features. From here, and unlike the previous situation, the error decrease is practically insignificant. As examples, the error difference from using 20 features to using 40 is a 0,8 % or a 0,55 % from 22 to 40.

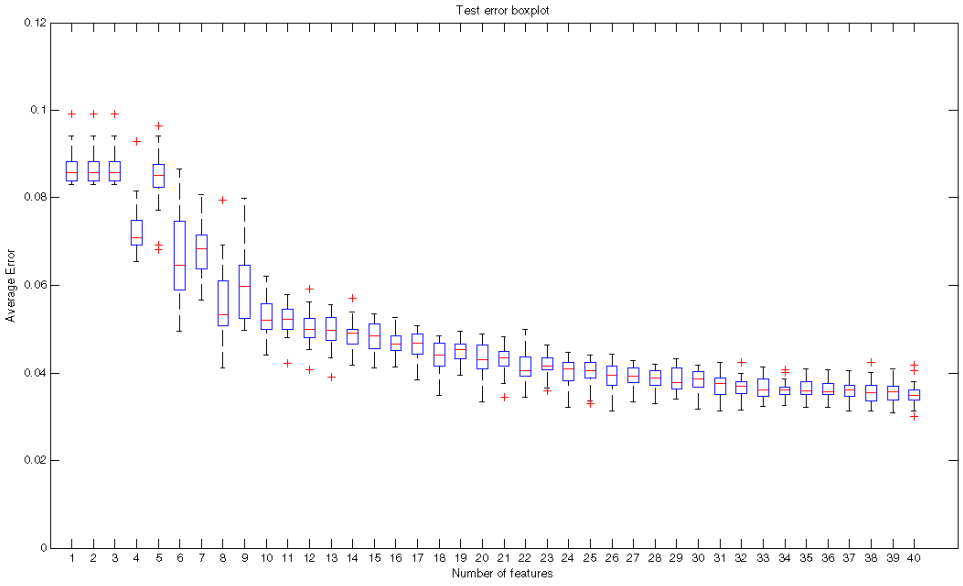

Figure 5.9: Statistical distribution of the test error curves as a function of the number of weak classifiers selected

5.3.2 Best features

The last step in the training process is to select one strong classifier with its corresponding weak learners. This strong classifier will be the one which will finally perform the speech/non-speech decision. By agreement, we choose the strong classifier that better performs in the test phase. That means that we implicitly choose the best features or dimensions in our 72-dimensional space. In other words, such features are those which best separate the data vectors from our training data and, as a consequence, achieve a lower test error. This does not mean that this classifier is the best for all the situations but for our training data.

The next figure shows the MS for a speech chunk with the best selected features. In white, the step or iteration at which the feature was selected (importance) is represented; in black, the number of the feature, from 1 to 72. For practical reasons, only the first 20 best features are represented. The complete list is found in A.1:

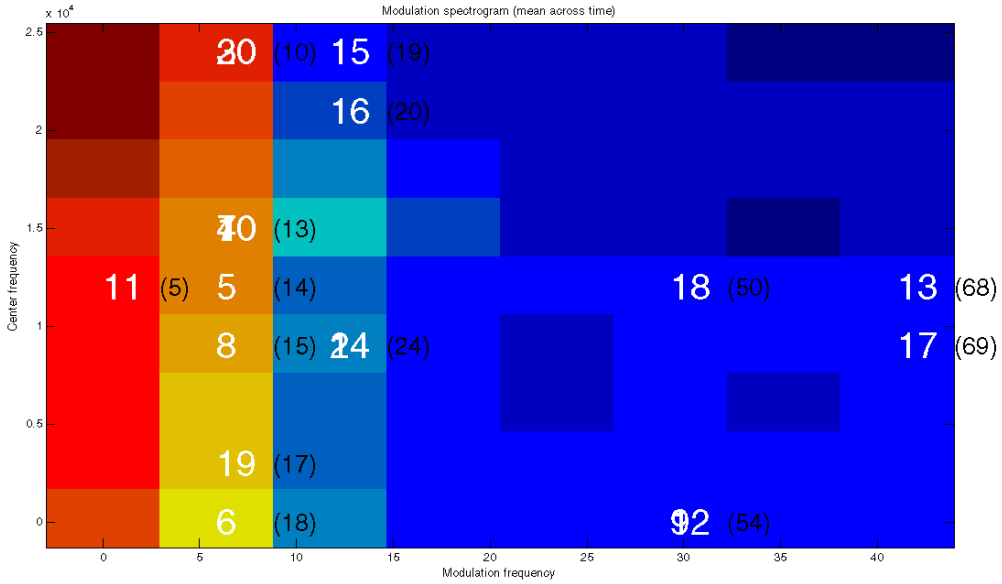

Figure 5.10: Best 20 features of the classifier used in this work

We can observe that from the best 20 features, 14 features are within the range 5-15 Hz, 5 above 15 Hz and only one below 5 Hz. That means that the most important features for the speech detection correspond to the modulations rates in the range 5-15 Hz, which corresponds to the results obtained by [7]. Furthermore, it also supports our initial hypothesis, that the third column in the MS, 10-15 Hz, should contain some distinguishing features.

5.4 Toolbox for acoustic event detection

5.4.1 General description

As it was said at the beginning, our goal was to build a toolbox for the acoustic event detection. At this point, we have already trained the classifier and adjusted its parameters from our data set. Furthermore, we have observed its performance on such data. Now, we would like to see the classifier's response to any input signal. To that end, we will use some audio files from

the *preliminary data set*, described in 5.1.2, and which haven't been used for the training phase.

This process is accomplished by means of a matlab routine, *AudioClassif.m*. It can be described as it follows:

$$[vecLabels, vecTimes, vecMgn] = AudioClassif(data, fs, data\_labels)$$

Given a digital audio signal, *data*, sampled at *fs* Hz, and optionally labeled, in *data_* labels, the function constructs an avi video, described in detail in Appendix C, and returns 3 outputs variables. A vector of labels, *vecLabels*, with the labels assigned by the classifier to each chunk. A vector of times, *vecTimes*, with the corresponding times to each chunk. And, finally, a vector of magnitudes, *vecMagn*. This last vector measures the level of speech or non-speech, with positive or negative values, respectively. The higher it is the magnitude, the more reliable becomes the classification. With respect to the video, it is formed by a representation of the magnitude through the time, with blue and red points representing speech and non-speech respectively. They are located at the times indicated in the *vecTimes* vector. The decision line is placed at the magnitude with value 0. On this line, and if the audio file was previously labeled, we can see thee false positives in cyan and the false negatives in magenta.

5.4.2 Experiments and results

In this section we will describe the results for two input signals corresponding to the two extreme situations: a speech situation (more_people1.wav) and a mixed situation, but with higher presence of non-speech sounds (multiple_cars2.wav).

The first signal, which has been manually labeled, corresponds to a continuous speech that takes place in a closed room. For that reason, most of the labels have been assigned to the "speech" class, whereas very few to the "non-speech" class. In particular, non-speech can be heard in the following time segments: 24.0-25.5 seconds, 35.5-37.5 seconds and 59.0-60.5 seconds.

The following figure shows a picture of the video created by the toolbox. It represents the time evolution of the speech/non-speech magnitude:

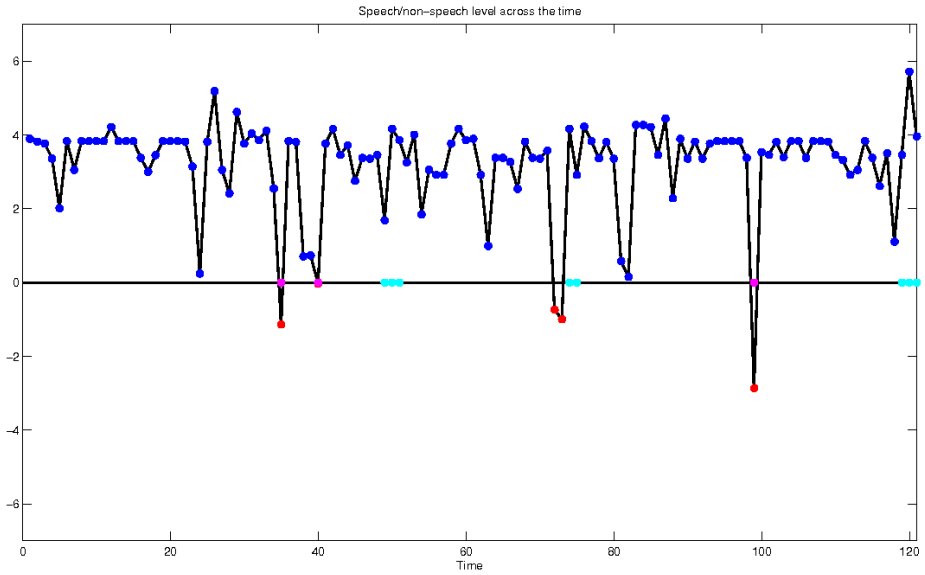

Figure 5.11: Speech signal classified by AB, with the misclassified points

As it was expected, most of the points have been assigned to the speech class. However, some misclassified points can be observed. The total error is about 9 % (11/121), which is much higher than the error achieved in the test phase (3.5 %) described in 5.3.

If we look at the figure attentively, we realize that most of the errors corresponds to false positives (6.6 %). That is, non-speech classified as speech. Then, if we listen to the audio file, we observe that the time segments 24.0-25.5 and 59.0-60.5 correspond to people laughing, whereas the segment 35.5-37.5 corresponds to silent. If we assume that for the training phase we used speech signals containing either people laughing or silent, then we can affirm that the classifier performed good and that the problem is due to a bad labeling of the data in the training and data set. As a consequence, if we don't consider such mistakes, now, the total error would be 2.5 % (3/121), which is below the average error obtained in the test phase. In addition and since the audio signal used to test the toolbox was manually labeled, some other mistakes could have been done. We can also observe two false positives close or on the decision line, which can difficult the decision.

The second experiment uses an audio signal with greater presence of car sounds. However, some speech segments are present and correspond to the

following time segments: 1.0-2.0 seconds, 10.5-11.5 seconds, 36.0-36.5 seconds and 39.0-39.5 seconds.

The Figure 5.12 shows the speech/non-speech level for each chunk. Remember that each chunk has a duration of 1 second, and as a consequence, if the current audio signal has a duration of 1 minute, then the number of chunks is equal to 120.

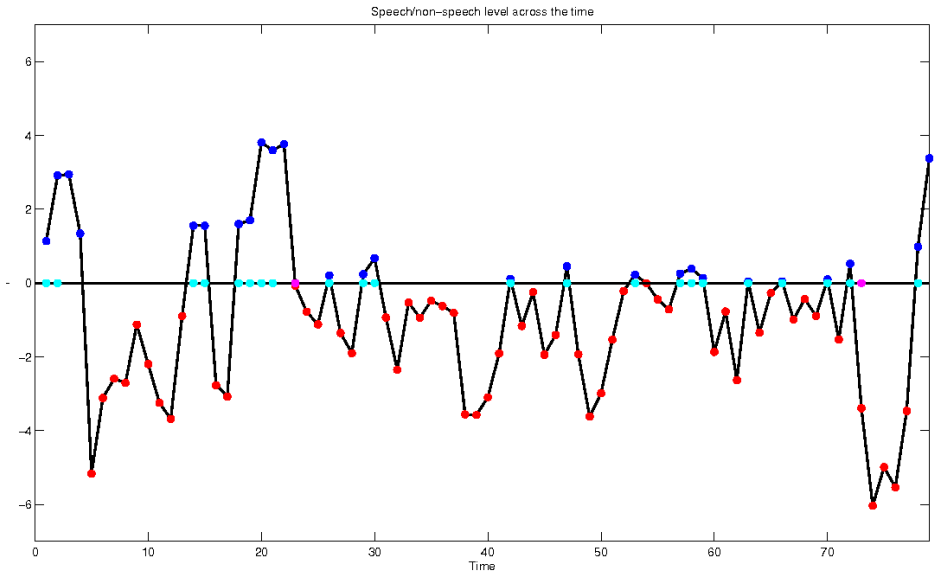

Figure 5.12: Car sounds signal classified by AB, with the misclassified points

Once more, most of the points have been assigned to the dominant class, non-speech. But now the performance of the classifier is worse, since the total error is much higher, approx. 30% (23/79). Again, almost errors correspond to fp (26 %).

If we observe the figure, we realize that most of the errors (15/23) are points very close to the decision boundary. When we listen to the audio file, the sounds that can be heard at such points or time instants don't correspond much to the sounds used as non-speech class for the training phase of the algorithm. Thus, the errors may come now for using very little representative sound samples for the non-speech class.

5.5 Comparison with related work

We will now use the software provided by [17][1] to compare our results, for the previous signals, with the results accomplished by their classifier. In A.2 are attached two tables showing the labels or classes assigned by the two classifiers for each chunk.

For the speech signal *more_people1.wav*, there is a difference between the labels of a 6.6 % (8/121). It is not strange to find so little difference between the two methods because the signal is speech (mainly) and both classifiers are trained with enough samples for this class. The Figure 5.13 shows the time instants where these differences occur:

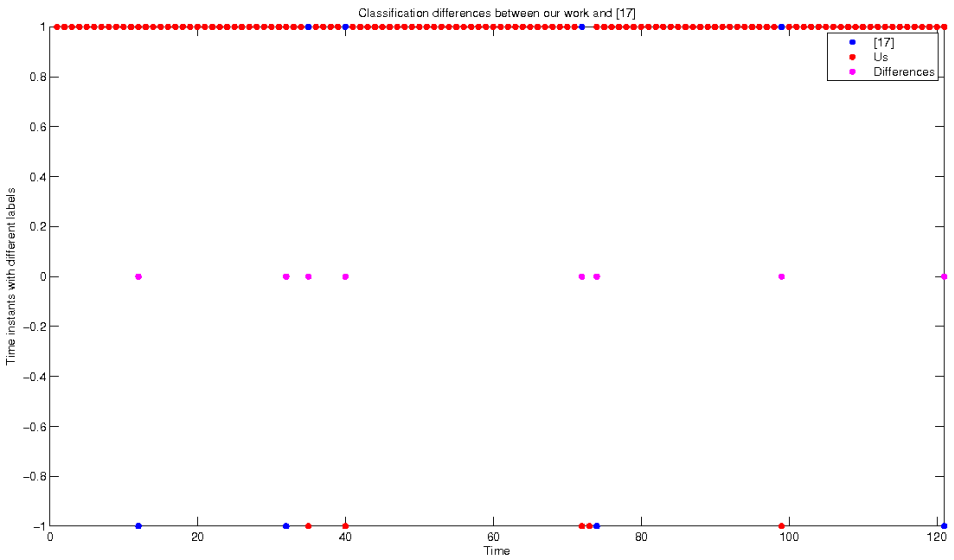

Figure 5.13: Classification differences between the two works for the signal *more_people1.wav*

On the contrary, the differences raise up to 30 % (24/78) for the sound *multiple_cars2.wav*. In this case, the task was more difficult due to the high variability for this class. There are so many sounds that can go into the

[1] DIRAC partner

non-speech class, that it is almost impossible to train the algorithm with all of them. The Figure 5.14 shows such differences:

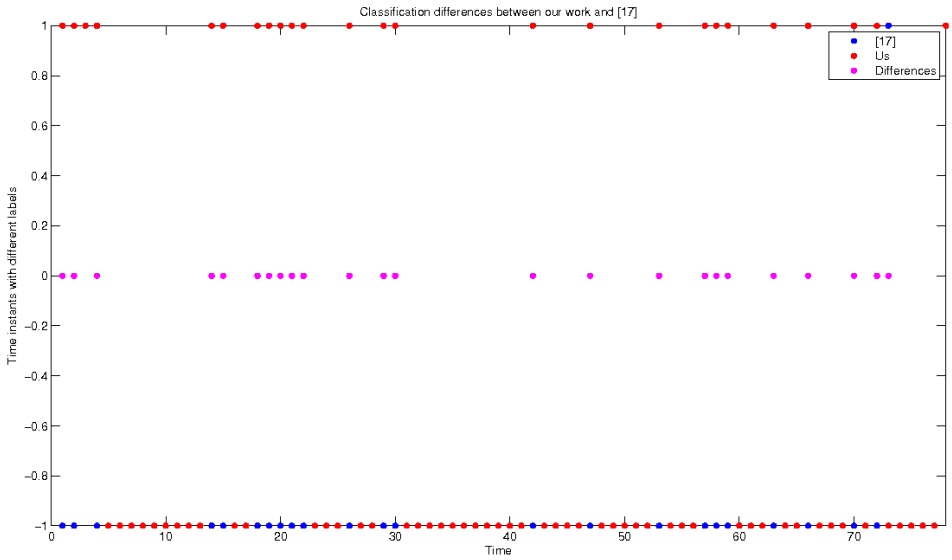

Figure 5.14: Classification differences between the two works for the signal *multiple_ cars2.wav*

As a conclusion, it has been observed that [17] and our classifier were comparable on speech data but on non-speech data [17] performed better since it yielded less false positives.

Chapter 6

Discussion and future work

6.1 Discussion

One of the main goals in this work was to observe the performance of a method never used before for the speech/non-speech detection, the adaptative boosting algorithm or AdaBoost.

We have found that a quite low error in the train and test phases can be obtained with a reasonable number of features. For the training data, a 4.5 % of the examples are misclassified using 12 features, a 3.5 % with 22 features or only a 2.4 % with 40 features. On the other hand, for the test data, as it was expected errors were just a little bit higher: 5 % when 12 features are used, 4 % with 22 features and 3.5 % with 40 features.

We have also confirmed our initial intuitions. The best and more distinguishing features for this task are found for modulation frequencies between 5 and 15 Hz, which is consistent with the results found in another works, such as in [7].

Finally, after observing the results from the toolbox, we have been able to draw some conclusions about the data used to adjust the parameters of the classifier. First of all, most of the errors in speech signals occur due to a bad labelling of the training data. If we listen to an speech signal and at the same time we observe how it has been classified, we see that some acoustic events, such as laughter or pauses, are assigned to the speech class. This is not strange if we assume that the training data used for that class contains this kind of events. In addition, other errors related to non-speech signals are due to an insufficient number of audio samples to represent this

class in the training data. There are many acoustic events that should be assigned to this class, but we have only trained the classifier with car sounds and environmental noise.

6.2 Future work

The first that should be done is to train the classifier with appropriate data sets. That is, correctly labeled samples for each class and an enough number of representative situations for both speech and non-speech.

Once we observe the results based on the new training data, we could redefine the weak classifiers to see if there is any improvement in the performance. In the present work we have used 1 dimensional classifiers or thresholds to build the strong classifier. Probably, using higher dimensional classifiers would entail a greater accuracy.

Appendix A

Tables

A.1 Best features

The next table represents the 40 best features selected from a analysis consisting in 20 data splits, each one with its corresponding strong classifier:

iteration	feature
1	19
2	24
3	10
4	13
5	14
6	18
7	13
8	15
9	54
10	13
11	5
12	54
13	68
14	24
15	19
16	20
17	69

18	50
19	17
20	10
21	11
22	26
23	15
24	27
25	20
26	49
27	59
28	15
29	14
30	64
31	66
32	5
33	11
34	18
35	8
36	10
37	24
38	59
39	19
40	3

Table A.2: Best 40 features of the strong classifier used in this work

A.2 Classification error

Here, we present two tables in order to compare the labels obtained by our classifier and the classifier from [17] for two acoustic situations. The first one mainly consists of speech; the second one of non-speech with some segments

of speech. The differences are emphasized in **red** and **bold**. The next table corresponds to the signal *more_people1.wav:*

#feature	**us**	**[17]**
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	1	1
7	1	1
8	1	1
9	1	1
10	1	1
11	1	1
12	**1**	**-1**
13	1	1
14	1	1
15	1	1
16	1	1
17	1	1
18	1	1
19	1	1
20	1	1
21	1	1
22	1	1
23	1	1
24	1	1
25	1	1
26	1	1
27	1	1
28	1	1
29	1	1
30	1	1
31	1	1
32	**1**	**-1**

33	1	1
34	1	1
35	**-1**	**1**
36	1	1
37	1	1
38	1	1
39	1	1
40	**-1**	**1**
41	1	1
42	1	1
43	1	1
44	1	1
45	1	1
46	1	1
47	1	1
48	1	1
49	1	1
50	1	1
51	1	1
52	1	1
53	1	1
54	1	1
55	1	1
56	1	1
57	1	1
58	1	1
59	1	1
60	1	1
61	1	1
62	1	1
63	1	1
64	1	1
65	1	1
66	1	1
67	1	1
68	1	1

69	1	1
70	1	1
71	1	1
72	**-1**	**1**
73	-1	-1
74	**1**	**-1**
75	1	1
76	1	1
77	1	1
78	1	1
79	1	1
80	1	1
81	1	1
82	1	1
83	1	1
84	1	1
85	1	1
86	1	1
87	1	1
88	1	1
89	1	1
90	1	1
91	1	1
92	1	1
93	1	1
94	1	1
95	1	1
96	1	1
97	1	1
98	1	1
99	**-1**	**1**
100	1	1
101	1	1
102	1	1
103	1	1
104	1	1

105	1	1
106	1	1
107	1	1
108	1	1
109	1	1
110	1	1
111	1	1
112	1	1
113	1	1
114	1	1
115	1	1
116	1	1
117	1	1
118	1	1
119	1	1
120	1	1
121	**1**	**-1**

Table A.4: Comparison between the labels obtained by our classifier and the labels obtained by [17] for the audio signal *more_people1.wav*

The table below corresponds to the audio file *multiple_cars2.wav*:

#features	**us**	[17]
1	**1**	**-1**
2	**1**	**-1**
3	1	1
4	**1**	**-1**
5	-1	-1
6	-1	-1
7	-1	-1
8	-1	-1

9	-1	-1
10	-1	-1
11	-1	-1
12	-1	-1
13	-1	-1
14	**1**	**-1**
15	**1**	**-1**
16	-1	-1
17	-1	-1
18	**1**	**-1**
19	**1**	**-1**
20	**1**	**-1**
21	**1**	**-1**
22	**1**	**-1**
23	-1	-1
24	-1	-1
25	-1	-1
26	**1**	**-1**
27	-1	-1
28	-1	-1
29	**1**	**-1**
30	**1**	**-1**
31	-1	-1
32	-1	-1
33	-1	-1
34	-1	-1
35	-1	-1
36	-1	-1
37	-1	-1
38	-1	-1
39	-1	-1
40	-1	-1
41	-1	-1
42	**1**	**-1**

43	-1	-1
44	-1	-1
45	-1	-1
46	-1	-1
47	**1**	**-1**
48	-1	-1
49	-1	-1
50	-1	-1
51	-1	-1
52	-1	-1
53	**1**	**-1**
54	-1	-1
55	-1	-1
56	-1	-1
57	**1**	**-1**
58	**1**	**-1**
59	**1**	**-1**
60	-1	-1
61	-1	-1
62	-1	-1
63	**1**	**-1**
64	-1	-1
65	-1	-1
66	**1**	**-1**
67	-1	-1
68	-1	-1
69	-1	-1
70	**1**	**-1**
71	-1	-1
72	**1**	**-1**
73	**-1**	**1**
74	-1	-1
75	-1	-1
76	-1	-1

77	-1	-1
78	1	1

Table A.6: Comparison between the labels obtained by our classifier and the labels obtained by [17] for the audio signal *multiple_ cars2.wav*

Appendix B

Program user's guide

B.1 Introduction

The goal of the program is to detect and classify some acoustic events. In particular, we want to be able to distinguish speech from other kind of sounds. In order to make the program more accessible to everyone, and to avoid understanding in detail each part of the code, we have developed a *graphical user interface* (GUI), which consists in two main menus.

The first menu is used to carry out the feature extraction, as well as some basic operations, such as load and save data. The second menu uses the values generated by the first one to perform, besides other tasks, the feature selection and the acoustic classification.

B.2 Menus description

B.2.1 Audiorecog menu

The matlab file audiorecog.m is the main program. It invokes the window that can be seen on Figure 2.1. This window has two columns. On the left, the labels indicate the operations that can be performed by the main program. In order to carry out any of these operations, the corresponding button on the right must be selected and the red button on the top, labelled as "A-RECOG", pushed.

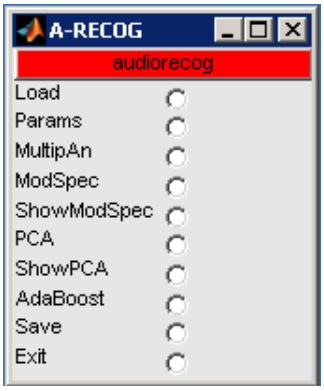

Figure B.1: Main program window. It performs feature extraction

The following lines give a broader description of all the operations available in this menu:

- **Load**: it loads previously saved data. We are asked to enter the name of the file to load (without extension) in the *Matlab Command Window.*
- **Params**: it allows the user to define the way in which the *modulation spectrogram* analysis is going to be performed. It can be done through the following graphical interface and consists in three options:
  - **Path**: only for single analysis. It is the name (with extension) of the wav audio file to analyse. It must be stored in the *Audio Signals* folder. See MultipAn to see how must the audio name be.
  - **TimeChunk** and **NumChunks** refers to the number of pieces in which we want to split the original audio signal. Each one of these chunks is going to be analysed separately. If a specific time resolution is desired, then the TimeChunk button must be pushed and the time, in seconds, must be entered in the box below. Its value by defect is 1 second. On the other hand, the NumChunks button must be pushed if we are not worried about the time length of each chunk, but in the total number of them. In spite of everything, they are two different ways to do the same thing.

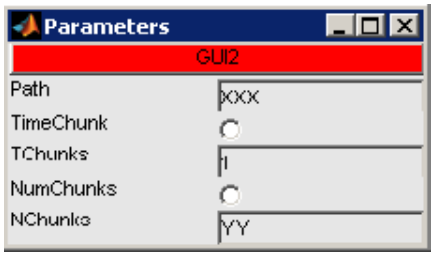

Figure B.2: The parameter menu allows to configure the analysis

- **MultipAn**: it performs the modulation spectrogram analysis of several files. They must be labelled and stored in the *Audio Signals* folder in the following way: "soundXX.wav", where XX is the number of the file. The labels file, "d.txt", must contain the class or category of each sound: 1 for speech sounds and -1 for non-speech sounds.

- **ModSpec**: it performs the modulation spectrogram analysis of a single sound. Its name must be entered in the Params menu. In this case, the label of the sound signal is not required.

- **ShowModSpec**: it shows the modulation spectrogram graphics of an audio signal or of its chunks. That means that a previous analysis must have been performed. On the contrary we are warned to do it. This option has been designed to show single file's graphics, so if it is selected after performing a multiple analysis, we will only see the first signal's graphics.

- **PCA**: it performs the *Principal Components Analysis* over one single file, indicated in the Params menu.

- **ShowPCA**: it shows the PCA graphic. Again, the PCA must have been previously performed.

- **AdaBoost**: it calls the second main menu, which will be explained in detail in the next section. It uses the data provided by MultipAn.

- **Save**: it saves the variable "variables" of the *Workspace* into a .mat file, with the name writen in the Path box, followed by "_out". For example, if the name in the Path is sound1.wav, the output file will

be "sound1_out.mat". This option can be used for single or multiple analysis.

- **Exit**: it exits the program.

B.2.2 ADB menu

The file xadb.m is invoked and the window shown in Figure 2.3 appears. It is an implementation of the AdaBoost algorithm, used in this work in order to distinguish speech from non-speech. The program uses the values obtained by the modulation spectrogram analysis performed over several files. To run any of the options, push the button beside the text label and the red button on the top.

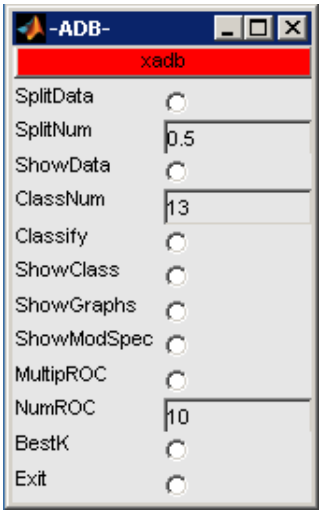

Figure B.3: The AdaBoost menu

A full description of the program's options is given below:

- **SplitData**: for single classification only. It splits randomly the available data vectors into two subsets: *train* and *test*.
- **SplitNum**: the number written in the box on the right is the percentage of vectors that will be assigned to the training set.

- **ShowData**: it shows the graphics of the data projections over each one of the 72 dimensions of the vectors. It is only for the training set.

- **ClassNum**: the maximum number of weak learners to be selected in order to build the strong classifier.

- **Classify**: for single classifications only (only one partition), it trains the AB algorithm and adjusts the parameters, given a training set. It shows the following graphics:

  - The first figure: in blue, the trained weak learners, and in green, the best ones or those that will be combined to build the strong classifier. Remember that each one of these best weak classifiers corresponds to a feature or coordinate in the 72-dimensional space. We can also see the training errors achieved by the strong and the first weak classifiers.
  - The second figure: the first weak classifier in green and the test errors achieved by the strong and the first weak classifiers.
  - The rest of the graphics show the data projections over the most distinguishing coordinates of the vectors and their corresponding weak classifier, for training and testing data. Absolute and relative errors are shown as well. This means that we can see the misclassified points on the graphic and the total training or test error.

- **ShowGraphs**: it shows the graphics of the ROC (false negatives vs false positives) and the total error as a function of the number of examples used, for one unique partition of the data.

- **ShowModSpec**: it shows one single modulation spectrogram graphic with labels on the most distinguishing features selected by the classifier. In white, it is represented the step in which the feature was selected (relative importance), and in black, the number of feature. This last number can go from 1 to 72, the maximum number of features of the data.

- **MultipROC**: it performs the classification process over different data partitions. Since each time a different data partition is used, different

error values are achieved. These are shown in the ROC and total error graphics all together.

- **NumROC**: it indicates the number of times that the task described in MultipROC must be done.
- **BestK**: it shows a graphic with different curves. Each one represents the training and test error achieved by the classifier for each data partition as a function of the number of features selected to build the strong classifier.
- **NumK**: it indicates the number of times that the analysis in BestK must be done. Once per each partition.
- **Exit**: goes back to the "audiorecog" menu.

Appendix C

Visualizaion of results

This appendix describes the process to go from an input data, previously labeled or not, to a video showing the chosen class for each chunk. It is accomplished by means of the matlab function *AudioClassif.m*, which was introduced in 5.4.1. It consists in:

1. Classification of the input signal, previously split into overlapping chunks of 1 second.
2. If the signal was labeled, then the routine calculates the classification errors by comparing both labels, the original one and the one provided by the classifier, for each chunk.
3. Create a sequence of frames to visualize the results. There are as many frames as chunks. Each frame consists in the following representations:
   (a) Vector of magnitudes, *vecMagn*, in black. It represents the evolution of the speech/non-speech level through the time.
   (b) The chosen classes for each time chunk. Speech in blue, and non-speech in red.
   (c) The classification errors, if they are available.
   (d) A vetical green line over the current chunk.
   (e) A number on the down left corner showing the current chunk.
4. Join the frames to build an avi video. Since we have 2 chunks per second, we hace chosen a *frame rate* of 2 fpm.

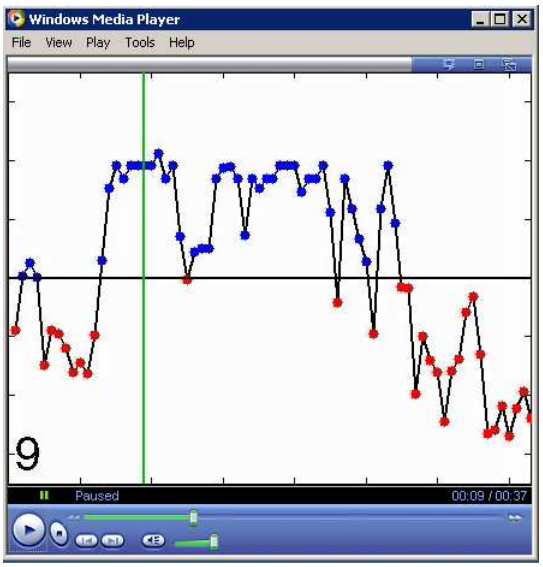

Figure C.1: Visualization of the classification results

Finally, if we play the video we will observe a line moving through the time, which indicates the class assigned by the classifier for the current time instant. Such time instant can be seen on the down left corner. An example is presented in Figure C.1:

Bibliography

[1] T. Arai, M. Pavel, H. Hermansky, and C. Avendano. Intelligibility of speech with filtered time trajectories of spectral envelopes. In *Proc. ICSLP '96*, volume 4, pages 2490–2493, Philadelphia, PA, 1996.

[2] Marios Athineos. `http://www.ee.columbia.edu/ marios/modspec/modcodec.html`.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Bussiness Media, New York, NY, 2006.

[4] Rob Drullman, Joost M. Festen, and Reinier Plomp. Effect of reducing slow temporal modulations on speech reception. *The Journal of the Acoustical Society of America*, 95(5):2670–2680, 1994.

[5] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. Wiley Interscience Publication. John Wiley, New York, 2nd edition, 2001.

[6] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[7] S. Greenberg. On the origins of speech intelligibility in the real world. In *Proceedings of the ESCA, workshop on robust speech recognition for unknown channels*, 1997.

[8] S. Greenberg and B. E. Kingsbury. The modulation spectrogram: In pursuit of an invariant representation of speech. In *Proc. ICASSP '97*, pages 1647–1650, Munich, Germany, 1997.

[9] Hynek Hermansky. The modulation spectrum in the automatic recognition of speech, 1997.

[10] Xuedong Huang and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001. Foreword By-Raj Reddy.

[11] Noboru Kanedera, Takayuki Arai, Hynek Hermansky, and Misha Pavel. On the importance of various modulation frequencies for speech recognition. In *Proc. Eurospeech '97*, pages 1079–1082, Rhodes, Greece, 1997.

[12] Birger Kollmeier and Rene Koch. Speech enhancement based on physiological and psychoacoustical models of modulation perception and binaural interaction. *The Journal of the Acoustical Society of America*, 95(3):1593–1602, 1994.

[13] Hari Krishna Maganti, Petr Motlicek, and Daniel Gatica-Perez. Unsupervised speech/non-speech detection for automatic speech recognition in meeting rooms. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2007.

[14] Peter Ladefoged. *A Course in Phonetics*, pages 165–196. Harcourt, Brace, Jovanovich, Orlando, 3rd edition, 1993.

[15] Peter Ladefoged. *Vowels and Consonants*, pages 133–153. Blackwell, Malden, MA, 2001.

[16] C. Nadeu, Dusan Macho, and J. Hernando. Time and frequency filtering of filter-bank energies for robust hmm speech recognition. *Speech Communication*, 34:93–114, 2001.

[17] Denny Schmidt and Joern Anemuller. Acoustic feature selection for speech detection based on amplitude modulation spectrograms. In *Fortschritte der Akustik*. DEGA, 2007.

[18] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book (Version 2.2)*. Entropic Ltd., Cambridge, 1999. `ftp://ftp.entropic.com/pub/htk/`.