Speech recognition system based on auditory features

Ignacio Martín Latorre
Departament de Teoria del Senyal
Universitat Politècnica de Catalunya

Written at Sony International (Europe) GmbH
Advanced Technology Center Stuttgart

from 1-5-2006 to 31-9-2006

Supervisor at UPC: Prof. Dr. Climent Nadeu
Supervisor at Sony: Dr. Franck Giron

Contents

Chapter 1

Introduction

1.1. Automatic Speech Recognition

Although automatic speech recognition (ASR) has undergone considerable improvements, the task of obtaining an accurate transcription from changes in the pressure of the air generated by the human speech production system is far from being complete.

The ability of communicating through speech has evolved in humans during thousands of years, yielding to the development of entire zones of the brain exclusively dedicated to speech production and understanding, and highly specialized organs that emit and decode speech signals in a huge range of environments. Children learn to speak and understand speech easily, and non impaired humans can decode speech production of new speakers without any training. As it is an innate ability of humans, it can be wrongly considered a trivial task, forgetting the fact that human language system is the result of a long evolution.

In spite of the intense research efforts made in achieving good recognition rates, nowadays artificial ASR systems are very far from the performance of humans in the same tasks. Although these systems work reasonably well in optimal conditions, when these conditions get worse the performance declines quickly, even if the system has been trained on the speaker's voice and a close-talk microphone is used to avoid noise and reverberation.

Standard approaches to automatic speech recognition (i.e. LPC) were initially built using production based processing schemes, considering the physical properties of sound pressure waves, and understanding the speech production as the result of an excitation signal applied to the input of the vocal tract.

Since long ago, these systems have been modified by incorporating perceptual features observed in physiological experiments. However, it is still

not clear which particular effects observed are crucial for the audition. Thus, experiments are conducted trying to emulate observed characteristics of the auditory system and conducting tests to measure the improvements in the overall ASR system. Quoting H. Hermansky, when humanity was trying to develop flying machines, they started by blindly imitate birds, flapping the wings, and finally they came to understand that the key was in Bernoulli's principle. Similarly, it is not clear which human characteristics should an ASR system try to emulate[6].

Speech can be considered as changes in the air pressure in time and frequency. Standard speech processing schemes, such as cepstral techniques, often use short term analysis frameworks, obtaining temporal sequence of frames of speech and analysing the frequential content of these frames. This content reflects the state of the vocal chords and the articulatory system. The assumption behind this procedure is that the these conditions do not change significantly between two frames. That is that the dynamics of the speech, the changes in time, will not be significant during the duration of such short frames. Part of the dynamics is captured by overlapping the temporal samples, but most of it is neglected.

In this work an automatic speech recognition system that incorporates a combination of emulation of the auditory system and spectro-temporal processing is studied. First, the speech signal is processed by the Seneff Auditory Model, inspired in collected physiological data. It, which includes information about the dynamics of the speech. Second, the resulting signal is filtered with a set of Gabor filters, a technique successfully used by M. Kleinschmidt that also tries to capture the dynamics of the speech, and was also motivated by physiological observations[16]. The choice of the set of Gabor filters is discussed. The motivations behind Seneff model and Gabor filters are completely different. In this project its combination is subjected to study.

Although the standard unit to be recognized in ASR is the phoneme, the usage of the Seneff Auditory Model with Gabor filters, a two-dimensional processing of the signal suggested the exploration of a different kind of unit, longer, that characterizes a group of phonemes pronounced in a row, that we named pseudosyllables. For performing this task we developed an algorithm to find these pseudosyllables and after that a procedure to change the initial labels of our existing speech database to use these different units that had been obtained by Viterbi segmentation using a previously trained HMM acoustic model. With the modified labels, the final ASR system can be trained.

The usage of Gabor filters presents another problem that was central in this study. As there is an infinite number of possible Gabor filters, we had to decide which ones where more useful to describe the signal and identify

the phonemic sequence. For this purpose, the multiclass boosting algorithm SAMME was implemented.

An ASR system has two main parts: feature extraction, where a vector of measures that should characterize the phonemic sequence is extracted from the signal, and pattern matching, where these features are processed using statistical methods to guess the phonemic sequence. Our work was focused in the first stage, but we will discuss the implications of the employment of these features in a standardASR system based on HMM.

We had at the laboratory an ASR system based in MFCC and HMM developed years ago by the Speech Group at Sony EuTec in the Janus Speech Recognition Toolkit [18], and a corpora described in chapter 3. Our practical work had three main parts: First, to develop Matlab scripts to Build the features filtering the Seneff signal with Gabor filters and write the SAMME algorithm in C++ to select the features; second, to evaluate a new type of segmentation of the signal in a unit that we called pseudosylables, and third, to debug the available ASR system, since when we tried to run it, it was not working.

1.2. Outline

This document is divided in 5 chapters.

Chapter 2 consists of a description of the models and algorithms employed both in the feature extraction process and in the overall speech recognition system.

Chapter 3 contains information about the software and corpora employed. As both have a influence on the results, their strengths and drawbacks are discussed.

Finally, chapter 4 contains a description of the results obtained at each stage, the process followed, and details about the implementation, as well as a discussion on the problems found.

Chapter 2

Theoretical description

The scheme of an ASR system can be viewed as a two-stage process.

First, we need to extract from the speech stream a set of measures that characterizes the phonemes. The goal is to discard redundant and useless information (dependent on the speaker or the environment), and acquire data that suffers little or no changes among utterances of the same phoneme but experiments changes when different phonemes are computed. In other words, the optimal set of features would contain information that depends only on the phoneme. This process is called feature extraction.

Second, the features acquired from the speech are fed into a system that performs a classification in order to determine the content of the speech. This task is done using probabilistic methods such as Gaussian Mixture Models and Hidden Markov Models, and receives the name of pattern matching.

This work deals mainly with the feature extraction part, and thus will be discussed in more detail.

2.1. Feature extraction

In order to achieve good recognition rates from a speech waveform, relevant information useful to identify the data and irrelevant information have to be separated. Feature extraction deals with this separation. Information that can help to identify the spoken phonemes should be emphasized, while information that depends on the irrelevant variations due to the particular pronunciation and the noise should be attenuated and ideally suppressed.

The process of acquiring these features can be done in multiple stages. The studied system starts with a waveform that in the first stage is processed with Seneff's auditory model, obtaining primary features. These features

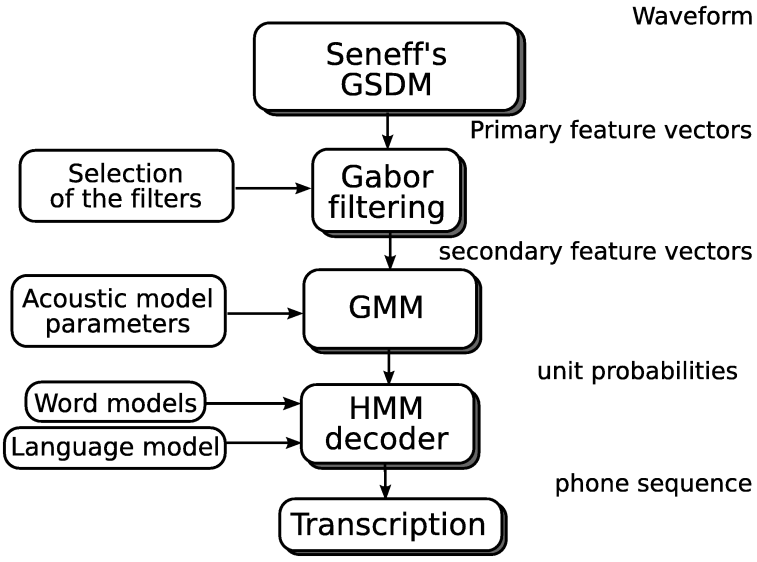

Figure 2.1: Diagram of the studied automatic speech recognition system from the waveform to its transcription .

are processed in a second stage by a bank of Gabor filters, which extracts information about the spectro-temporal evolution of the signal, obtaining secondary features.

2.1.1. Auditory front-end

The question of the idoneity of the imitation of the auditory system for ASR has been subject of discussion[6]. Nevertheless it is clear that both human speech production and recognition systems have evolved to adapt each other. As the long term goal of an ASR system is to emulate human speech recognition performance, it makes sense to emulate also the human auditory perception, although it is difficult to identify the crucial tasks that the human auditory system performs. There are several auditory models that try to reproduce specific effects observed in empirical results. In our work, we used the Seneff's joint Synchrony/Mean-Rate Model.

Seneff's joint synchrony/mean-rate model

The auditory model used in this work is Seneff's Joint Synchrony/Mean-Rate Model, proposed by S. Seneff in 1988[19]. This model was motivated by physiological properties observed in the auditory system of mammals. The properties matched by this system are considered relevant to the processing of the speech, and it was found that the system can emulate some

physiological experiments. We did not write the code for this model, it was already available at the laboratory.

The model, described in the block diagram of Figure 2.2 can be decomposed in three main stages.

The first two stages model the peripheral transformations occurring in the first stages of the human auditory system. The third stage deals with perception of the signal.

Critical band filter bank First, the signal is band-limited to 6.5 kHz and sampled at 16 kHz. Then, it is convolved with a cascade of 40 complex high-frequency zero pairs filters, with steep cutoff in the high-frequency side and broad low-frequency tails. Frequency region between 1.5 and 5.0 Khz is emphasized 10-20 dB, boost justified by experimental results that stated that the outer ear resonances have the effect of increasing the energy in this region[22]. This filterbank is designed to match physiological data obtained about the vibrations of cat's basilar membranes[3]. A plot of the responses of these filters is shown in Figure 2.3. The number of 40 channels was chosen to represent the signal, being enough to show some detail while avoiding excessive complexity that would result in higher computational time. As this decision was taken in 1990, maybe now more channels could be employed.

Inner-hair-cell/synapse model The second stage, which name is Inner-hair-cell/synapse model, is intended to model the transition from basilar membrane vibration to probabilistic responses of the auditory nerve fibers. Its output represents the probability of firing of the nerve at a given moment. It is composed by a half-wave rectification stage, a short-term adaptation circuit, a low-pass filter, and a rapid automatic gain control. Each of these components are nonlinear, except for the low-pass filters. Each one is associated to a particular effect observed in physiological experiments. The ordering, important due to the non-linear nature of the models, is partially justified by associations with elements of the auditory system and the sequence in which they are placed in animals.

The half wave rectifier is modeled mathematically as:

$$y = \begin{cases} 1 + Atan^{-1}Bx & \text{if} \quad x > 0 \\ e^{ABx} & \text{if} \quad x \leq 0 \end{cases}$$

Where B sets the operating range of the channel. It also models the auditory feature: the bandwith of the response at intermediate amplitudes is higher than at higher amplitudes, where saturation appears.

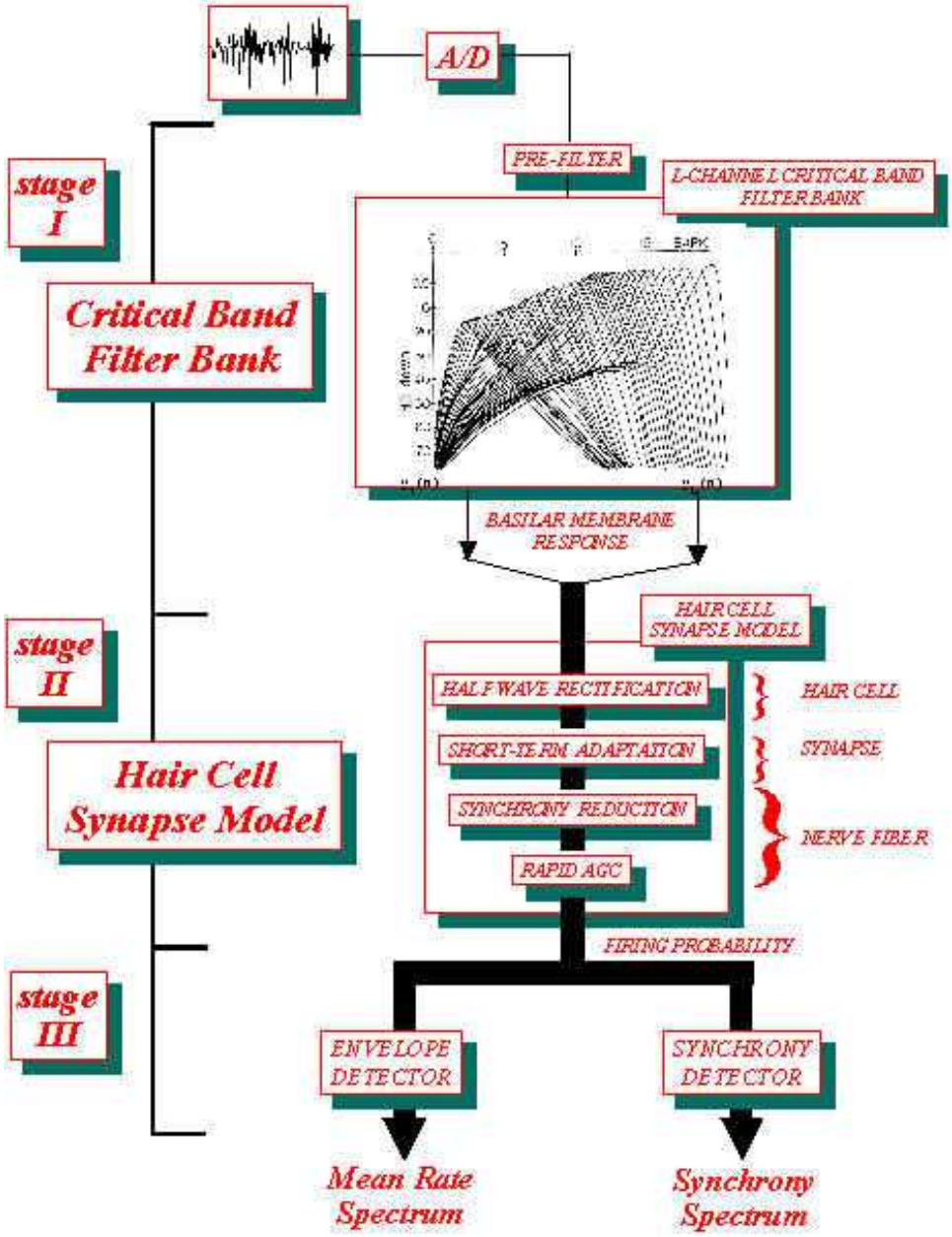

Figure 2.2: Block diagram of the Seneff's Joint Synchrony/Mean-Rate Auditory Speech Processing. From [2].

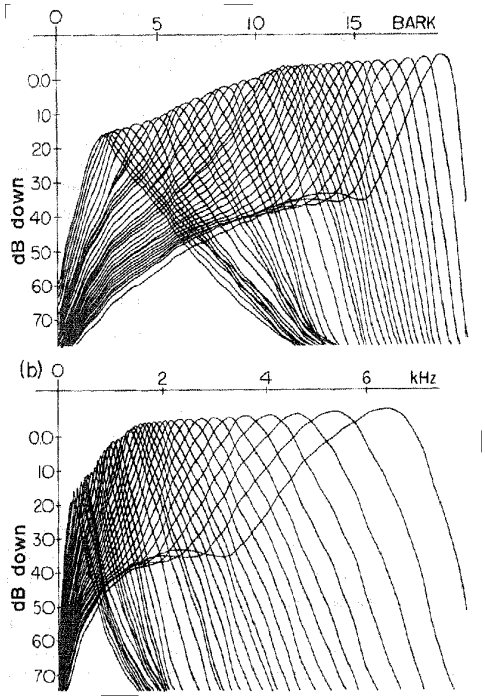

Figure 2.3: Frequency response of the filterbank, in Bark scale (a) and in linear frequency scale (b). From [19].

The short term adaptation circuit is intended to model the synaptic region between the hair and the nerve fiber. Mathematically is defined as:

$$dC(t)/dt = \begin{cases} \mu_a[S(t) - C(t)] - \mu_b C(t) & \text{if} \quad C(t) < S(t) \\ -\mu_b C(t) & \text{if} \quad C(t) \geq S(t) \end{cases}$$

The channels in this membrane are disabled when the concentration gradient of a substance (a neurotransmitter or an ion) is too small, and proportional to the concentration gradient when it is higher than a certain value, being μ_a the proportionality constant. Also, the substance is naturally lost in a rate proportional to μ_b. The output of this model is $\mu_a[S(t) - C(t)]$, that stands for the flow of the substance across the membrane and is related to the probability of firing as a function of time. It models observed auditory effects. In response to steep high-amplitude signals the output is high, and then decreases with time constant $\tau_1 = \frac{1}{\mu_b}$. If the signal decays abruptly, the output will be zero until the decay of $C(t)$ (decay that obeys to an exponential with constant $\tau_2 = \mu_b$). τ_2 plays an important role modelling the forward masking effect. Forward masking occurs when the response to sound is attenuated as it is preceded by a signal of greater energy.

The low-pass filter models the loss of synchrony in nerve-fiber responses and other locations in the auditory system. It reduces the synchrony with high frequencies. The transfer function of the filter is:

$$H(z) = \left(\frac{1 - \alpha}{1 - \alpha z^{-1}} \right)^{n_{LP}}$$

Where n_{LP} is the order of the filter, composed by integrators with time constant τ_{LP} and α being the location of the pole on the real axis of the z-plane.

The automatic gain control models the refractory effect of nerve fibers, and is defined as:

$$y[n] = \frac{x[n]}{1 + K_{AGC} E(x[n])}$$

It reproduces the auditory response against onsets, and alters deeply the envelope of the signal.

Generalized synchrony detector model (GSDM)

Finally, the output is processed in order to analyse the synchrony of the model. The original GSD model that Seneff proposed can be described as the quocient of sum of the signal and the same signal delayed divided by the difference of the signal with its delayed version

$$z[n] = \frac{y[n] + y[n + d]}{y[n] - y[n + d]}$$

Thus, if the signal has as period equal to the delay value in the algorithm, the output of the system is high. Each channel has a different delay. Nevertheless, the input of the system studied in this work was not produced by such method, but by a different method property of Sony that we used as a black box, without knowing its details.

In Figure 2.4 is provided a graphical representation of an utterance processed with the GSDM system and MFCC.

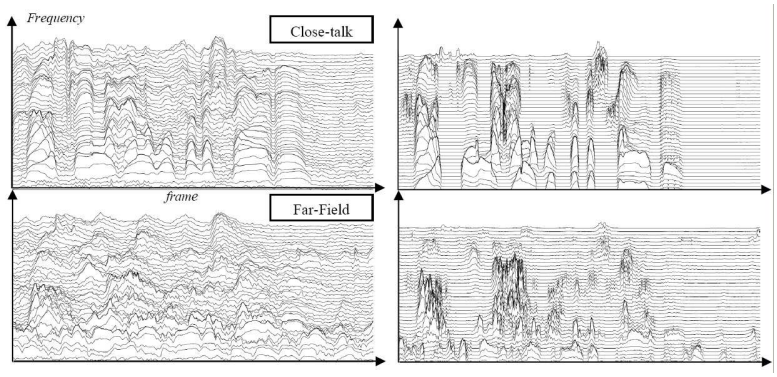

Figure 2.4: MFCC representation (left) compared to GSDM representation in two different environments. The figures on the top show a utterance recorded with a close-talk microphone, while for the figures on the bottom a studio microphone placed a 3m of the speaker was used. The speech was produced in a reverberant room in the presence of an air-conditioner. From [5].

2.1.2. Spectro-temporal processing

Speech is characterized by the evolution of the signal both in time and frequency. An usual approach to speech recognition is to take temporal frames and analyse their spectral characteristics. In these techniques the temporal evolution of the spectrum of the signal is not taken into account very much. For overcoming these limitations, there is a number of methods that have been proposed, such as delta and delta-delta features or RASTA, or TRAPS, where a number of multi-layer perceptrons are trained with the temporal evolution of spectral energy in Mel-scale along different bands.

Michael Kleinschmidt proposed a new approach to this problem based on the utilization of localized spectro-temporal filters (LSTFs)[16]. This was motivated by physiological experiments with mammals that showed that a large percentage of neurones in the primary auditory cortex have different responses when the auditory system is excited with upward-moving and downward-moving ripples in frequency. Based on this evidence, Klein-

schmidt looked for a method able to extract these patterns, and proposed the use of Gabor filters as spectro-temporal filters.

Gabor filters

The filtering is performed by computing the correlation of each filter, represented by a matrix of values, with the two-dimensional signal resulting from Seneff's model. The output of this process is a new bidimensional signal.

The complex Gabor function proposed in [16] is the product of a complex sinusoidal function with a Gaussian envelope. The envelope has a width determined by its deviations σ_f and σ_t, and the frequency of the sinusoid by ω_f and ω_t. The two independent parameters of the spectral and temporal frequencies allow the adaptation of the Gabor filter to particular orientations in frequency and time of the modulation.

$$Gf(t,f) = g(t,f)s(t,f)$$

where $g(t,f)$ is the Gaussian envelope

$$g(t,f) = \frac{1}{2\pi\sigma_t\sigma_f} e^{\frac{-(t-t_0)^2}{2\sigma_t^2} + \frac{(f-f_0)^2}{2\sigma_f^2}}$$

and $s(t,f)$ is the sinusoidal function

$$s(t,f) = e^{i\omega_f(f-f_0)+i\omega_t(t-t_0)}$$

Where f_0 and t_0 determine the center of the Gabor function. The infinite extension of the Gaussian envelope is limited to $1.5\sigma_t$ from the center. In this work we employed the absolute value of the response and, since the number of potential Gabor filters is huge, it was limited at first to only round-shaped filters ($\sigma_t = \sigma_f$).

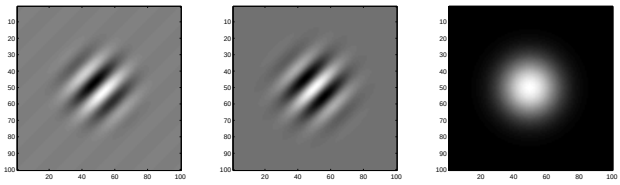

Figure 2.5: A Gabor filter. From left to right: real part, imaginary part, and absolute value.

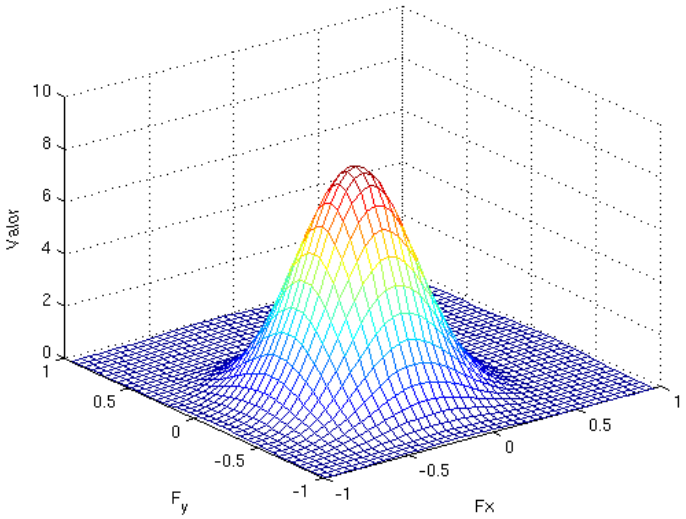

Figure 2.6: Frequencial response of a complex Gabor filter.

2.1.3. Feature selection

While we expect Gabor filters to extract relevant information from the signal, there still remains the question of which set is the best for the classification task.

The output of each Gabor filter should have the following properties:

- To be different between samples of different classes.
- To be similar between samples of the same original class.

Ideally, if a single Gabor filter fulfills these requirements, the problem would be successfully solved. Unfortunately, there is not such a filter for the problem of identifying phonemes or pseudosyllablic units. For that reason, we will call Gabor filters a *weak* classifier. Instead, we can try to find a combination of Gabor filters able to maximize those criteria when working as a whole. That is, to employ the best finite set of Gabor filters that we can find, and use its outputs as a postprocessing of the original features that will serve us better to achieve the goal of differentiate among the phonemes.

But then again, as the number of Gabor filters is infinite, evaluate all the possible combinations of sets of Gabor filters is an impossible task. For automatically selecting the set that produces the best features for classifying *as a whole*, we need an algorithm that evaluates different sets of *weak* classifiers "to emit a final decision better than the individual decisions of

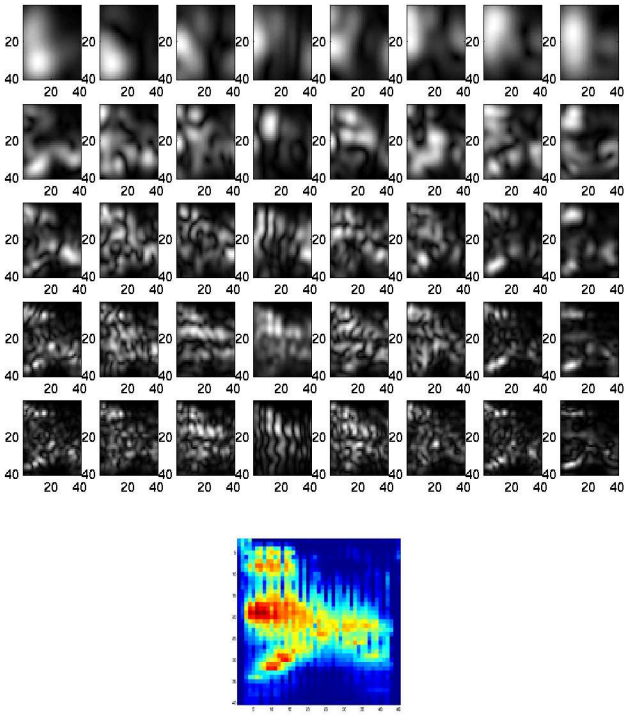

Figure 2.7: Filtering example: Pseudosyllable /ar/ in a GSDM representation (bottom, in color) filtered with 40 Gabor filters with frequencies ranging from 1/10 cycles per point to 1/2 cycles per point. Lower frequencies are on the top, and produce blurrier results. The higher are in the bottom and produce noisier signals. Orientations are rotated in steps of 25.5 degrees.

each one". Moreover, we want the algorithm to tell us which classifiers are the most reliable, working as a whole. The approach used in this project is SAMME, a boosting algorithm.

Boosting

Research in boosting algorithms has been conducted since long time. Probably the most simple case is democracy: we have a set of classifiers that *vote*. Formally, we have a function $h(\mathbf{x})$ that, from an input $\mathbf{x}$, returns a discrete result $\{h : \mathbb{R}^N \rightarrow \{1...N\}\}$. The final decision is the most frequently chosen class.

However, although this algorithm can be useful, there have been further improvements in this field. Kearns and Valiant [8][7] were the first in raising the question of whether *weak* learning algorithms that perform better than random guessing can be combined in a *strong* learning algorithm with arbitrary precision.

In 1995, Freund and Schaphire proposed the AdaBoost algorithm[4]. AdaBoost (that stands for ADAptive BOOSTing) has proven to be successful in producing accurate classifiers when applied to two-class classification problems. Among other advantages, it is particularly resilient to overfitting, and Schapire and Freund have shown classification problems where the test error continues decaying even after the training error arrives to zero.

AdaBoost takes as input a training set $(\mathbf{x_1}, c_1) \ldots (\mathbf{x_n}, c_n)$, where $\mathbf{x_i} \in \mathbb{R}^p$ are observations of a random variable and c_i is integer e.g. $\in 1, \ldots, K$ that denotes the class of which the variable $\mathbf{x}$ is an observation. The goal is to find a classification rule $C(\mathbf{x})$ such as, given a new input $\mathbf{x}$, the algorithm assigns it a class label c and achieves the lowest possible misclassification rate in new samples.

AdaBoost takes an iterative approach to solve this problem. For M iterations, it calls a *weak learning algorithm* that *weakly* classifies the given data. The only requirement for these algorithms is to be able to achieve a classification error better than 1/2 (in two-class problems, better than random choice). In order to improve the overall error, the algorithm is adaptive. One of the main ideas behind AdaBoost is to modify a list of weights $w_1, \ldots, w_n$ for the training set along the iterations. While initially all weights are set to the same value, when a data point is misclassified its weight is increased (boosted). This has the effect that in the following iteration the weak learning algorithm will pay more attention to the problematic data. At each iteration, a weight $\alpha(m)$ is assigned to the trained weak learning algorithm as a measure of its reliability. At the end, a *strong* classifier is build by evaluating all the weighted weak learners. Figure 2.8 shows the

evolution of the weights in a two-class classification problem, and Figure 2.9 shows the evolution of the strong classifier.

Specifically, if $T^{(m)}$ is a weak multi-class classifier, and $\mathbb{I}(A)$ has value 1 when the expression A is true and 0 when A is false, the algorithm AdaBoost proceeds as follows:

AdaBoost algorithm Freund and Schaphire, 1997

1. Initialize the observation weights $w_i = 1/n, i = 1, 2 \dots n$
2. For $m = 1$ to M:
   a) Fit a classifier $T^{(m)}$ to the training data using weights w_i.
   b) Compute
   $$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}(c_i \neq T^{(m)}(\mathbf{x_i}))/\sum_{i=1}^{n} w_i$$
   c) Compute
   $$\alpha^{(m)} = log\frac{1 - err^{(m)}}{err^{(m)}}$$
   d) Set
   $$w_i \leftarrow w_i exp(\alpha^{(m)} \mathbb{I}(c_i \neq Y^{(m)}(\mathbf{x_i}))), i = 1, 2, \dots, n$$
   e) Re-normalize w_i
3. Output:
   $$C(\mathbf{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \mathbb{I}(T^{(m)}(\mathbf{x}) = k)$$

AdaBoost has been proved to be extremely successful in two-class classification problems, but it has some drawbacks in the case of multiclass problems. Although there have appeared [4] generalizations of AdaBoost for multiclass problems, as AdaBoost.M1, they impose the same constraint to the weak learners choice: they must have an accuracy greater than 50%, otherwise, if $err(m) > 0.5$, $\alpha(m)$ will take negative values, and the weights will be updated in the wrong direction. This limitation, that in two-class problems is easy to overcome (the weak learner should be better than random choice), as the number of classes grows, becomes more restrictive. For instance, in a 100 class classification problem and equally distributed data, random guessing gives an accuracy of 1%, so the weak learner is expected to perform 50 times better than random choice.

Hence, AdaBoost may fail if the weak classifiers are not accurate enough, and this depends on the number of different classes[23].

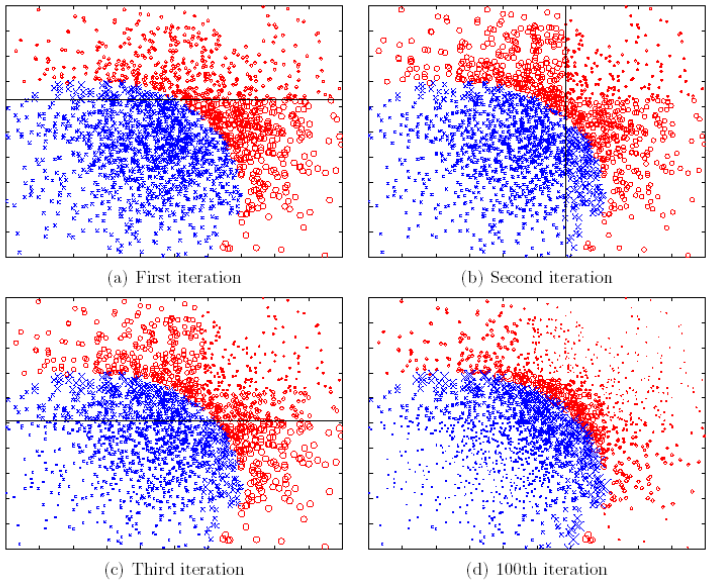

Figure 2.8: Weights evolution during the iterative process of AdaBoost algorithm. Weights are represented as the size of the markers of the samples. After 100 iterations, the weight is concentrated in the samples of the boundaries between classes. From [1].

There is another way to solve this problem while still using AdaBoost, that consists in converting multi-class problems into a couple of two-class problems. Two popular approaches are the *one vs. rest* scheme, where we define the first class as a class of the original problem and the second as the set of all the remaining classes of the original problem, and the *pairwise* scheme, where each time only two classes are considered.

In terms of computational complexity, if the weak learner used is a classification tree, the depth of each tree is d and p is the dimension of the input, then the computational cost for constructing each tree is $O(dpn\log(n))$. Then if M is the number of iterations and K the number of classes, then the computational cost of both the pairwise and the one vs. the rest schemes is $O(dpn\log(n)MK)$.

Thus, as K grows, AdaBoost becomes more and more expensive. In our experiments, with 100 classes, this became a big issue to consider that made this approach prohibitive.

SAMME algorithm

Recently, a new multi-class classification algorithm has appeared with the name of SAMME[23]. The algorithm is very closely related to AdaBoost:

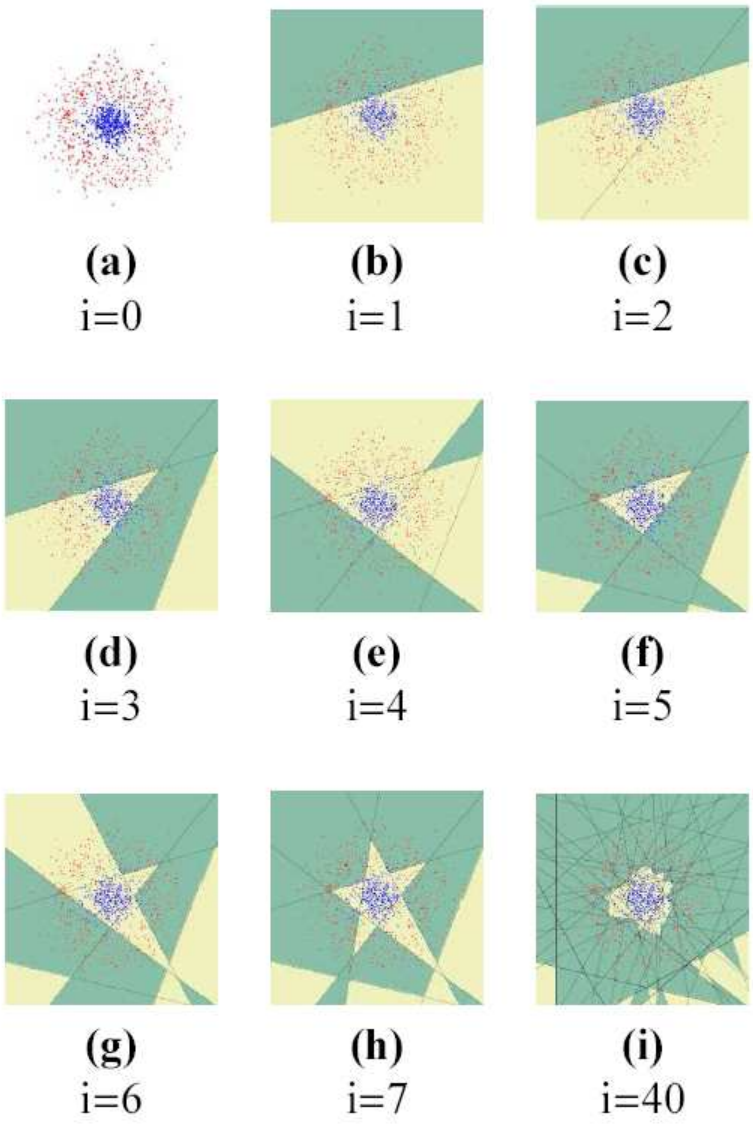

Figure 2.9: Evolution of the strong classifier built by AdaBoost according to the number of iterations i. From [13].

SAMME algorithm Freund and Schaphire, 1997

1. Initialize the observation weights $w_i = 1/n, i = 1, 2 \dots n$
2. For $m = 1$ to M:
   a) Fit a classifier $T^{(m)}((x))$ to the training data using weights w_i.
   b) Compute
   $$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}(c_i \neq T^{(m)}(\mathbf{x_i}))/\sum_{i=1}^{n} w_i$$
   c) Compute
   $$\alpha^{(m)} = log\frac{1 - err^{(m)}}{err^{(m)}} + \log(K-1)$$
   d) Set
   $$w_i \leftarrow w_i exp(\alpha^{(m)} \mathbb{I}(c_i \neq Y^{(m)}(\mathbf{x_i}))), i = 1, 2, \dots, n$$
   e) Re-normalize w_i
3. Output:
   $$C(\mathbf{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \mathbb{I}(T^{(m)}(\mathbf{x}) = k)$$

It can be noticed that the only difference between SAMME and AdaBoost is the computation of $\alpha^{(m)}$. SAMME adds a new term $\log(K-1)$, that depends on the number of classes (notice also that if $K = 2$, SAMME is the same as AdaBoost with two classes). This term is obtained considering the two-class AdaBoost algorithm as a forward stagwise modeling, and generalizing its loss function to the multi-class problem. For further details see [23].

Compared with AdaBoost, SAMME only imposes to the weak learners the requirement of having less error than $1/K$. It is enough if they perform slightly better than random guessing, independently of the number of classes. In our case, with 100 classes, this means that our weak learners can be 50 times less accurate.

This time, in terms of computationaly complexity, if the weak learner used is a classification tree, the depth of each tree is d and p is the dimension of the input, M is the number of iterations and K the number of classes, then the computational cost of SAMME is $O(dpn\log(n)M)$. This makes SAMME K ($K = 100$ in our experiments) times faster than a multi-class AdaBoost classification reduced to K two-class AdaBoost classifications.

Apart from this consideration, SAMME shares with AdaBoost the rest of its features and strengths. In particular, it is very resistant against overfitting.

These are the reasons why we choose this algorithm, that we implemented in C++.

2.1.4. Mel-frequency cepstrum coefficients

The system under study was intended to be compared with the widely used Mel-Frequency Cepstrum Coefficients (MFCC) approach to feature extraction. We had a MFCC system developed with the Janus Speech Recognition Toolkit that had been tested years ago in the same task (word and short sentence recognition of the same corpora). Unfortunately, when we run the MFCC, the system was not working as should, a long stage of debugging was needed to make it work again. This debugging task was not theoretical, but a practical work with the TCL scripts written for Janus. Nevertheless, as we used this system widely and for comparison with the new methods studied, we provide here a short description of the MFCC features.

The process to compute MFCC has three stages: first, the signal is decomposed in temporal frames; second, it is processed through a Mel-filter bank; third, then the cepstral coefficients are computed.

An additional advantage of using MFCC is that they are highly uncorrelated, and thus, diagonal covariance can be assumed. This assumption becomes very useful when using Gaussian Mixture Models (GMM) for training the Hidden Markov Models (HMM).

This method can be considered a frequencial approach to feature extraction, because the temporal dynamics of the signal are not strongly influential on the results, as each frame is considered isolatedly. This represents a crucial difference in comparison with the spectro-temporal analysis performed with Gabor filters. On the other hand, it has been studied and used extensively during the time, becoming one of the most popular methods for feature extraction on the ASR field.

Furthermore, it is important to notice that the only component of the whole process motivated by analysis of the properties of the human auditory system are the Mel-filter bank. This represents also a big contrast when confronted with Seneff's auditory models features postprocessed with Gabor filters, methods strongly inspired in physiological properties of the auditory system.

From a third point of view, MFCC approaches for feature extraction are much less complex in terms of computation. Only the processing with Seneff's auditory model is more expensive than the whole MFCC process. Gabor filtering also introduces an extra non-negligible computational time.

Windows

First, the signal is splitted in temporal intervals or frames. These frames have typically a length ranging betwen 10 ms and 50 ms. This is because of the assumption that the vocal tract will not experiment big changes in its shape in short intervals, and thus the signal can be considered quasi-stationary (this assumption is true with most of the phonemes, but not with plosives, for instance). Moreover, it should be big enough to be possible to perform analysis in the frequency domain with reasonable accuracy.

In order to preserve temporal changes, usually the frames overlap. Thus, the distance between the start of consecutive frames is not equal to the frame length, and receives the name of frame shift. The value of these frame shifts is usually between 5 ms and 20 ms.

Although rectangular window has good frequency resolution, it presents small attenuation on the side lobes. Thus, usually another window is applied to the frames, often the Hamming window:

$$w[n] = 0.54 - 0.46cos\left(\frac{2\pi n}{N-1}\right)$$

Mel-filter bank

The FFT of the windowed signal is calculated, and then is processed thorough a set of filters known as Mel-filter bank. These filters are placed according to the mel scale, that gives more importance to sounds with lower frequencies, in an attempt to emulate perceptual characteristics of the human auditory system[21]. The filters are distributed at the same distance in the Mel scale, which was determined experimentally:

$$Mel(f) = 2595\log_{10}\left(1 + \frac{f}{700}\right)$$

The outputs of these M filters $H_m[k]$ form a vector $Y[m]$:

$$Y[m] = \sum_{k=0}^{N-1}\left(|X[k]|^2 H_m[k]\right) \qquad 0 \leq m < M$$

.

Cepstrum

From $Y[m]$ is then calculated the logarithm of the signal:

$$S[m] = ln(Y[m]) \qquad 0 \leq m < M$$

The operation is completed by computing either the Inverse FFT or the Discrete Cosinus Transformation (DCT) of $S[m]$.

$$c[n] = \sum_{m=0}^{M-1} S[m] cos\left(\frac{\pi n(m-1/2)}{M}\right) \qquad 0 < n \leq M$$

The first coefficients of $c[n]$, associated with low frequency components, depend on the shape of the vocal tract, while higher order coefficients are associated with the excitation of the vocal chords.

The number of cepstral coefficients is a parameter to set. Values between 13 and 39 are often used. As reported in [14], high order coefficients (1-15) contain information about the fine structure of the spectrum (high frequency), and low order coefficients about the smoothed spectrum (low frequency(15-40)). It is also useful to discard the first coefficient, that contains a measure of the signal energy in the frame, value with high variability that can depend on multiple factors.

2.2. Pattern matching

The features extracted are then feeded to an acoustic model. The model employed is Gaussian Mixture Models (GMMs), which provides a set of probabilities for each phoneme, with Hidden Markov Model (HMM) that performs a search for the most likely sequence of phonemes employing statistic information from a training task and information about the language provided by the language model.

We did not write any of these modules, that were already implemented in Janus, although we debugged the scripts to isolate and solve the bugs present in the code.

2.2.1. Acoustic model based on hidden Markov models (HMMs)

While GMMs provide the likelihoods of the presence of each phoneme or segment of phoneme in a given time, its combination with Hidden Markov Models (HMM) allow to consider the time dynamics of speech, dealing with the problem of estimating the most likely sequence of units. This approach has been widely used in speech recognition.

If we consider a system that can has a state s from a set of N different states. Considering the time dimension as discrete, with intervals of Δt, the system can change its state (or remain in the same state) according to probabilities related to the state. This system receives the name of first order Markov chain if the state at a given time q_t only depends on the previous state s_{t-1}:

$$a_{ij} = p(q_t = s_t | q_{t-1} = s_i, q_{t-2} = s_k \ldots) = p(q_t = s_t | q_{t-1} = s_{t-1})$$

And has a probability π_k of the first state being s_k.

This model, in contrast with Hidden Markov models, can be called observable, since the output of the process at a given time is the state q_t at that moment. However, the speech cannot be modeled as an observable Markov model, because we do not have direct observations of the changing states in the mind of the speaker while he is producing the speech. This process is hidden, but we can obtain observations of another stochastic process (the speech signal) that is bounded with the sequence of states of the hidden process. This additional process emits an observation O_t from a set of K (v_k) different observations that depend only on the current state q_t:

$$b_{jk} = p(O_t = v_k | q_t = s_j)$$

The hidden Markov model is determined by the parameters

$$\lambda = (\pi, A, B)$$

Being $A = [a_{ij}]_{N \times N}$ and $B = [b_{jk}]_{N \times K}$

Speech usually depends on more than one of the previous states. However, HMM models are widely and successfully used in speech recognition.

Given this definition, there are three problems that have to be solved to use the models in automatic speech recognition:

- **Evaluation problem** How to compute the probability of an observation given the model $P(O|\lambda)$?
- **Decoding problem** For a given sequence $O = O_1 O_2 \ldots$ and a model λ how to find the sequence $Q = q_1 q_2 \ldots$ that best explains the observations O?
- **Learning problem** How to ajust the parameters $\lambda = (\pi, A, B)$ such that the distribution $P(O|\lambda)$ corresponds to the events modeled?

The first problem can be solved by calculating every possible observation sequence. Given that there are N^T sequences, this method is too compley, with order $O(N^T)$.

Fortunately there are affordable solutions to this problem, as the forward algorithm, of complexity $O(N^2T)$.

In the second problem, the Viterbi algorithm, with complexity $O(N^2T)$, was used for performing a search of the state sequence that maximizes $P(O|\lambda)$.

The third problem is related with the training. There is no analytical method that maximizes $P(O|\lambda)$ by adjusting the parameters. Then, it was used an iterative method, the Baum-Welch or forward-backward algorithm.

These three methods are explained in detail in [17].

There still remains the question of how to estimate the probabilities B. Usually, the output distribution probability is modeled by a mixture of multi- variate normal distributions, method that receives the name of Gaussian Mix- ture Models (GMM), that will be explained here, since these models are of great importance when studying new features.

2.2.2. Gaussian mixture models (GMMs)

Gaussian mixture models are used to model a probability density function by describing it as a sum of Gaussians.

If b_{jk} is the probability density function of an observation O_t being produced by an state s_j

$$b_{jk} = b_j(v_k) = p(O_t = v_k|q_t = s_j)$$

then it can be expressed by arbitrary precision as a mixture:

$$b_j(v_k) = \sum_{i=1}^{N} c_{ji} b_{ji}(v_k)$$

Where each component density is a n-variate Gaussian function:

$$b_{jik} = b_{ji}(v_k) = \frac{1}{(2\pi)^{(n/2)}|\sum_{ji}|^{1/2}} exp\{-\frac{1}{2}(x-\mu_{ji})\Sigma_{ji}^{-1}(x-\mu_{ji})\}$$

GMM constraints

An important fact about GMM is that, although they can approximate, in theory, a probability density function with arbitrary precision, some problems arise in real world cases, due mainly to computational complexity constraints:

- The number of gaussians is limited.
- Usually, Σ_{ji} is assumed to be diagonal.

The consequences of the first constraint depend strongly on the shape of the probability density function that has to be approximated. This effect is particularly bad when modeling distributions that present steep peaks, because they get masked by the gaussians.

The second constraint is widely used in MFCC-based systems because it is a good approximation of the covariance of MFCC features and the computational complexity is considerably reduced, because the number of parameters for each Gaussian becomes N instead of $N(N-1)/2$. However, this assumption is not always true, and has great importance when dealing with new features, such as the studied Seneff-Gabor features, as will be discussed later.

2.2.3. Language model

Language models incorporate knowledge about the statistics of sequences of phonemes or other units. For example, the may provide the probability of occurrence of a given phoneme given the previous phonemes.

N-grams models simplify the problem by assuming that the probability of occurrence of a phone p_k given the $1 \dots k$ previous phonemes depends only on the N previous phonemes.

$$P(p_k|p_{k-1} \dots p_1) = P(p_k|p_{k-1} \dots p_{k-N})$$

They are a useful way to incorporate information from the N nearest neighbours to the estimation of p_k. By their nature, these models become dependent on the training data. If an N-gram does not appear in the training data, either its probability of occurrence will be zero, or smoothing techniques should be applied to the computation of the probabilities in order to avoid an excessive dependence on the training dataset. As N increases, the number of possible N-grams increases exponentially, needing larger datasets and consequently increasing the dependence on the training dataset. In our experiments N was set to 3, therefore we used a tri-phone language model.

In the problem studied, we worked with isolated word recognition. The words of the set were compiled in a dictionary along with their different possible pronunciations, and the recognizer was restricted to produce only words from the dictionary, thus simplifying the task.

Chapter 3

Working environment

3.1. Corpus

Corpora are collections of speech recorded material. They are used for training and evaluating automatic speech recognition systems. The statistical models will be subsequently adapted to the patterns of these utterances in order to model correctly similar data. The choice of the corpus should be related to the task that the ASR should perform.

In our experiments, we used the German corpora at Sony Eutec.

3.1.1. Composition

The corpus used in our experiments contains a long set of recorded utterances from 100 different German speakers, both male and female, in absence of noise. It is composed of recordings of few words (mostly one or two), covering a list of different topics:

- Adresses. 100 per speaker. Example: Beethoven Platz
- Letters. 26 per speaker. Example: a
- Cardinal numbers. 50 per speaker. Example: 1
- Dates. 20 per speaker. Example: Montag
- Frequencies. 20 per speaker. Example: hunderteins Komma drei
- Commands. 350 per speaker. Example: überspringen
- Ordinal numbers. 30 per speaker. Example: erster
- 4 digit pin numbers. 50 per speaker. Example: null sieben null neun

- Phrases. 30 per speaker. Example: Radio bitte etwas leiser
- Radio stations. 100 per speaker. Example: WDR 4
- TV stations. 50 per speaker. Example: Premiere
- Telephone numbers. 50 per speaker. Example: null null acht eins drei vier sieben acht zwei drei eins
- Teletext numbers. 50 per speaker. Example: dreihunderteinundvierzig
- Times. 50 per speaker. Example: Vier Minuten und vierzig Sekunden
- Sequences of letters (A..Z). 100 per speaker. Example: A I B O
- Sequences of digits (0..9). 100 per speaker. Example: drei drei drei

Giving a total of 1176 recorded utterances per speaker. 25 speakers were used for training, and the remaining 75 for testing.

3.2. Labeling

A corpus can be labelled, which means that there is available information about the time at which the unit starts and ends. Units in this context can mean words, phones, or subphones. Labeled corpora are very useful for our purpose of training an ASR system. Unfortunately, labeling large corpora is a very time consuming (and computationally expensive) task.

A suboptimal solution for this problem is to label the corpus with a previously trained ASR system, and force it, knowing the transcription, to align the labels with the time domain signal in the most likely way using the Viterbi algorithm. This is how our corpus was aligned to phoneme strings. But, as current ASR systems are far from perfect, the alignment produced may be very inaccurate. Again, the importance of these misalignments depends also on the statistical model that is being trained with the copora.

In order to illustrate this, take the following example. In Figure 3.1, the phoneme /b/ is clearly misaligned. /b/, being a plosive phoneme, is articulated by obstructing totally the air flow, and later releasing the obstruction to produce a small plosion, which creates its characteristic sound. That means that /b/ can be described as a silence preceded with a particular plosion. If the phonemes are modelled by Markov chains with three states -b -m -e (that stand for beginning, middle, and end) as in 3.2, ideally state -b will be the steep transition to silence, state -m will be the silence, and -e the again steep plosion of the sound. When modeled with this kind of topology, it is not very important how long these segments are, because each

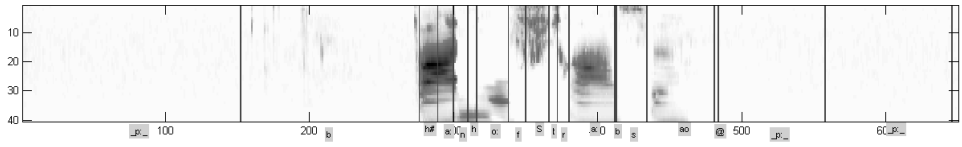

Figure 3.1: Labeled Seneff's auditory system output of the word Bahnhofstrasse.

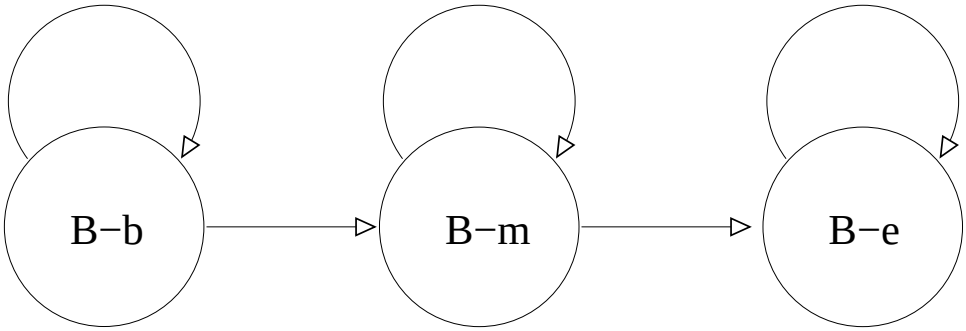

Figure 3.2: Topology of the Markov model of the phoneme /b/ in three subphonemic units.

Markov state have an arc that allows the system to stay in that determined state. Thus, in these models, these types of misalignments are not crucial.

However, if the approach changes, misalignments can play a significant role, as we will see in the next chapter when moving from phonemes to pseudosyllables.

Chapter 4

Experimental work

The work conducted in this study has three main parts.

First first place, the particular characteristics of the features resulting from the application of Seneff's Auditory Model and their postprocessing with two dimensional filters raised the question of whether we should try to recognize phonemes or other longer basic units. In section 4.1, an algorithm to re-segment the signal in a different kind of units is discussed, as well as its outcome.

Second, the usage of Gabor filters needs some kind of selection process to select the filters that best characterize these basic units. A boosting algorithm was implemented to perform this task, and its results are discussed.

Third, once we have the secondary features, there is still the question of whether they suit or not a standard pattern matching scheme with Gaussian Mixture Models and Hidden Markov Models.

4.1. Resegmentation in pseudosyllablic units

Having choosen the processing stages of the recognizer, there still remains the question of what units should the system work with. Standard systems recognize words or phonemes, depending on the task.

The speech streams in our database were already segmented in phonemes. That task was already performed automatically using an ASR system available at the laboratory. In this process, speech is segmented by finding the most likely sequence using the Viterbi algorithm given the transcription. Each sound datafile in the database is processed with this method and the outcome is a file that specifies the segments of speech that correspond to each phoneme. The information that associates a segment with a phoneme receives the name of label.

However, as the Seneff's GSDM auditory model output presents groups of shapes clearly separated, may be more natural to recognize these shapes, that are sometimes single phonemes and sometimes groups of phonemes, that we called pseudosyllablic units instead of phonemes.

This resegmentation involves two tasks. First, we have to find these shapes and their boundaries, and then the existing labels need to be modified. The criterion behind the placing of the boundaries was to look for valleys in the energy of the signal, following the assumption that this method would separate groups of phonemes pronounced together. Once we had the boundaries, then we could modify the labels of the corpus, moving the boundaries of the phonemes and grouping them in longer units. That was done to solve misalignment problems.

The process can be described as follows: The values of the 40 channels of the Seneff's model output are accumulated at each time frame, being the resulting value the sum of all 40 channels. That transforms the two-dimensional signal in a single dimensional vector along time. After normalizing the signal dividing it by its maximum value, it is smoothed by convolving it with the Hann window:

$$w[n] = 0.1\left(1 - cos\left(\frac{2\pi n}{N-1}\right)\right)$$

The value of N was adjusted empirically to 40.

After computing the logarithm of the vector to have a less peaky and smoother representation, local maxima and minima are extracted. Local minima can be indicative of the presence of a valley in the signal energy. However, not every local minimum separates two different shapes. A shape may contain local minima, that should be discarded.

To discard irrelevant minima, from the set of local minima and maxima, we looked for sequences of maximum-minimum-maximum wider than 140 degrees. This discards slow variations of the signal energy. Values of maximum-minimum-maximum with a difference of less than 0.5 dB were also discarded, eliminating false boundaries, as shown in Figure 4.1.

To compute the angle in sequences of maximum-minimum-maximum the dot product was used:

$$\theta = \arccos\left(\frac{a \cdot b}{|a||b|}\right)$$

where a and b are vectors defined in Fig 4.1.

This technique, although very simple, performed reasonably well when detecting the valleys without other information apart from the energy of

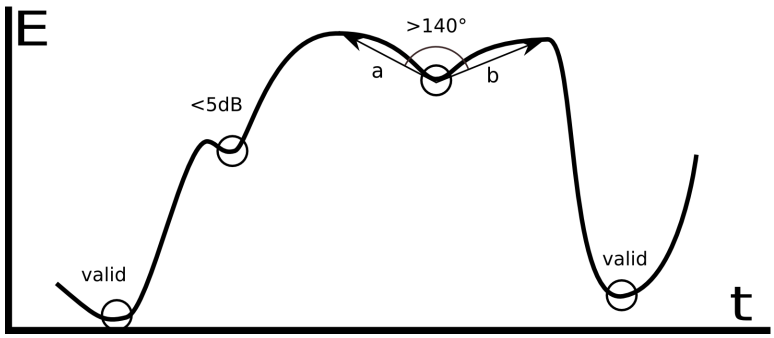

Figure 4.1: Diagram showing the rules for discarding local minima.

the speech signal. It has, of course, limitations. For instance, eventual noise with more than 0.5 dB of energy difference with respect to the silence will not be discarded.

We had another kind of information that should be used to re-segment the speech signal, the automatically aligned labels, that are sometimes completely wrong, and sometimes inaccurate, but most of the time can help to discard wrong results.

The first approach to the goal of assigning phonemic labels and merge them in pseudosyllabic units was to compute the boundaries of the segment finding relevant valleys with the method previously explained, and then assign the automatically aligned boundaries of the phonemes with the nearest pseudosyllabic boundaries. This produced wrong results. Short phonemes can, for instance, disappear if their start and end boundaries are assigned to the same point.

The method finally employed focused in overlaps instead of distance between boundaries. The process, depicted in Figure 4.2 was done in two stages. First, the phonemes were assigned to the region of the new boundaries they overlap more with, and the assignation was stored. If two or more phonemes share the same boundaries, they are merged in a pseudosyllable. Second, some segments were not assigned after the first stage, resulting in gaps without labeling. If that happens, we look for the phoneme that they overlap more with, and assign the gap to the same pseudosyllable to which that phoneme was previously assigned. In Figure 4.2 , the gap is assigned to the blue phoneme, but as it was assigned in the first stage with the first pseudosyllable, the gap is merged with the first pseudosyllablic unit, increasing its length.

Segments labeled as silence were not merged with other phonemes, and in consequence, no pseudosyllable contained silence segments.

This process was acceptably accurate. When later we looked at the list of assignations for the same pseudosyillabic unit, we found that the shapes

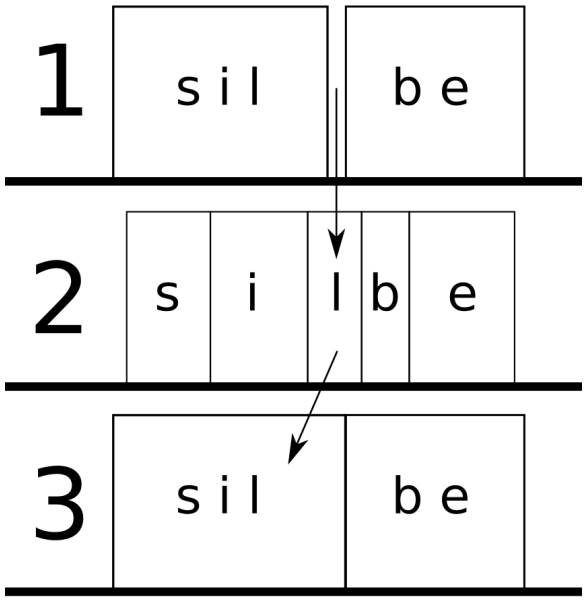

Figure 4.2: Diagram showing the process of gap assignation. After the first stage, the phonemes of German word *silbe* have been assigned to two pseudo-syllables, but there is a gap between them (1). We look at the automatically assigned phonemes to see which phoneme overlaps more with the gap, and it is /l/ (2). As this phoneme has been assigned to the first pseudosyllable, we assign the gap to the first pseudosyllable too.

obtained that had the same labeling, had similar aspect as well.

However, this procedure has its drawbacks. Although the same units looked similar, the total number of units was very large.

Placing the boundaries in valleys in the signal has a problem: although it works with many phonemes, there are some whose signal contains a valley. This happens mostly with plosive phonemes, like /t/, that is characterized by a short silence when the air is obstructed and then a plosion when the airflow is freed. When smoothing the signal by convolving it with the Hann window, the signal is smoothed, a valley can change its place. Hopefully, it will be *inside* the boundaries of the original /t/. But, as we are segmenting according to valleys, even in that case the phoneme has a 50% probability of being assigned to the segment that does not contain the plosion. The problem is better explained with diagrams, and it is shown in Figure 4.3. Although this particular case can be solved simply assigning the plosives to the phoneme pseudosyllable that follows them in time, it shows the type of problems that can be expected from this approach.

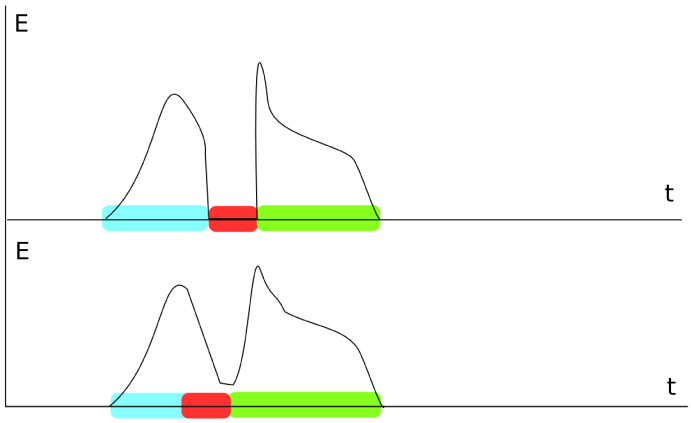

Figure 4.3: Scheme showing a typical problem of the realignment. After smoothing with a Hann window and finding the valleys, a /t/ (in red) is assigned to the pseudosyillabe of its right, which does not contain the plosion, only the silence.

This kind of problems increased the number of different pseudosyillabes in the set, multiplying its number by adding randomly half of the times problematic units at the start or at the end of the pseudosyllables.

However, the biggest problems were produced by other factors, mainly two:

First, the particular articulation of the speaker can produce valleys in the signal in unusual places, and the algorithm has no way to overcome these

difficulties. Sometimes it is fixed in posterior stages of the algorithm, but sometimes these wrong valleys are selected as boundaries by some phonemes, forming the boundaries of a pseudosyllable. The effect of these artifacts is to increase the number of different pseudosyllables in a non desired way.

Second, we relied strongly on the accuracy of the previous alignment, because that is the only information we have about its phonemic composition. As has been said, the labels contained errors very often, errors that were carried along the process. Sometimes, they were fixed by the algorithm, but many of them persisted.

In particular, the labeling of short words was particularly wrong. The corpus contained several utterances with isolated numbers or letters. The phonemes in these utterances had been wrongly automatically segmented, and almost always all the phonemes were assigned to segments of silence, instead of speech. That is to say, misalignments were very frequent in short utterances in the labels existingat the laboratory. When we processed these utterances with this technique, as silence does not contain energy valleys, so regions with silence are considered a pseudosyllable. As in these utterances all the automatically assigned phonemes overlapped with a region of silence, all of them were assigned to a single, long pseudosyllable whose signal was a segment of silence.

To overcome these alignment problems, some simple rules were applied in order to discard utterances with obvious bad labeling. However, visual inspection of all the utterances would had been prohibitive due to the large size of the training and test databases. The rules applied were:

- If there is a phoneme whose duration is 350 ms or more, the utterance is discarded. As the length of each frame was 5 ms and they did not overlap, that means a duration of 70 frames. If after the process a resulting pseudosyllable has a length of more than 70 frames, the utterance is discarded too. That is to say, the rule of 70 frames is applied before and after the process. These long units are very likely result of bad automatic alignment in the training database. This rule was not applied to the silence, since it is usual to have long segments of silence.

- If a resulting pseudosyllable is composed by more than 4 phonemes the utterance is discarded.

- If the utterance has less than 5 phonemes it is discarded because wrong labellings were too frequent in this kind of utterances.

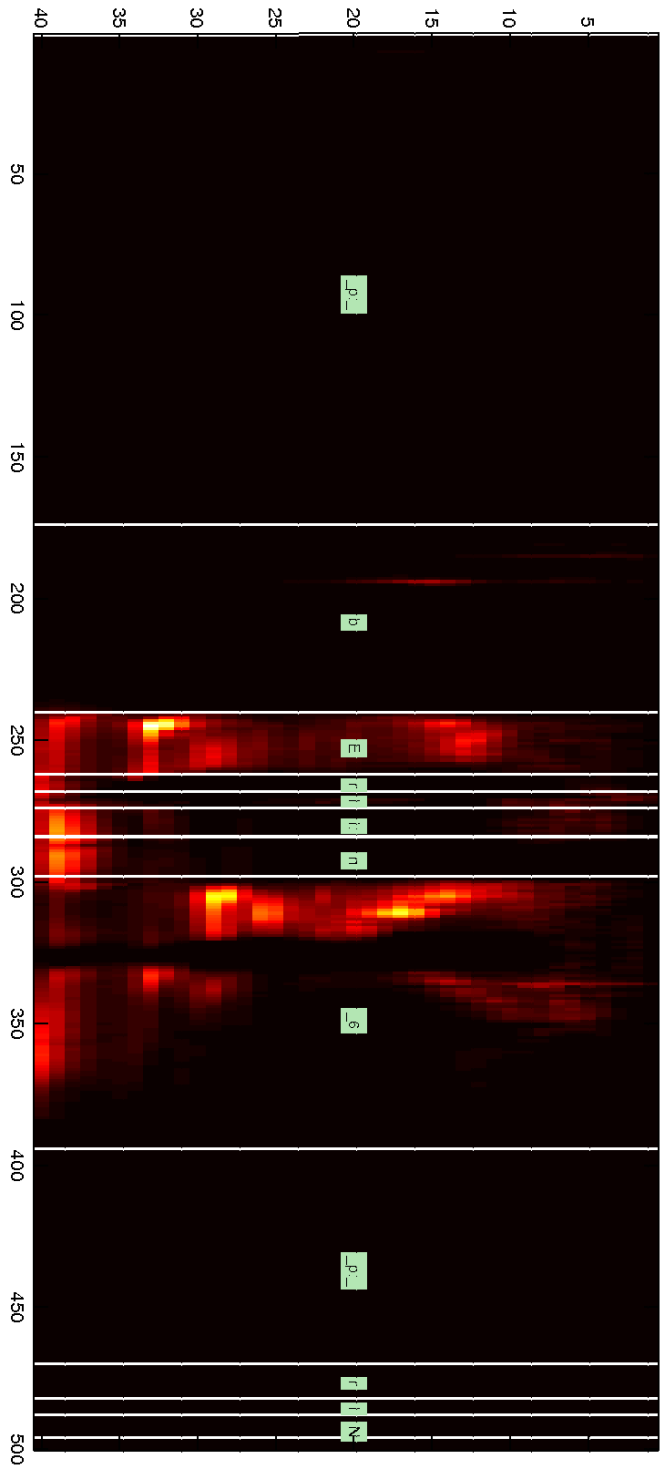

Figure 4.4: Utterance *Berliner Ring* illustrating a typical discarded utterance because of its bad initial labeling.

4.1.1. Reduced set of pseudosyllables

After relabeling every utterance in the database with the process above described, the number of different pseudosyillabes (more than 5000) was too large to perform a classification task. Moreover, most of the units appeared only a few times in the database, making the training process difficult because of the lack of a sufficient number of training samples per unit.

In consequence, we computed the number of utterances that could be expressed with a given set of units, expecting that, after a reasonable number of pseudosyllables in the set, most of the database could be expressed with this subset. We started from the 50 most frequent units, adding new units according to its frequency of occurrence in the database.

However, the results where not as good as expected. The number of utterances that could be expressed with a reasonable number of pseudosyllables was quite low. An acceptable number of pseudosyllables that could express most of the corpus did not exist, at least with the information available and the algorithms employed. To express most of the corpus with pseudosyllables, a very large number of pseudosyllables would had been necessary, but that means a large number of classes to classify both in the Gabor selection and in the speech recognition processes.

Moreover, as the number of pseudosyllabes grows, there are more and more pseudosyllables very similar, making the classification task difficult. For instance, many of the pseudosyillabes started with the phoneme /t/, very frequent in German (in fact, the third most frequent isolated phone of the list). Considering that productions of /t/ are usually short phonemes composed from a silence and a plosion and they were attached to a longer phoneme, and given that the plosion can be wrongly assigned to the previous pseudosyllable, the result is that often pseudosyllables starting with t had only a silence representing the /t/. Given, for instance, /ta/, was very likely that it consisted of a silence and the /a/, being thus very similar to an isolated /a/, specially if it is an /a/ that appeared in the beginning of a word. As SAMME focuses in samples that are difficult to classify, it can be expected the selection of the Gabor filters to be affected by the attempts of finding Gabor filters able to discriminate between, for instance, /a/ and /ta/, both very frequent and frequently indistinguishable in our framework.

The number of pseudosyillabic units was finally limited to 100 units, an amount that, although is large enough to bring difficulties to the classifiers, is not excessive. As commented before, there were no reasons to set the limit in any particular number. These 100 units included single phonemes, being thus possible to express the whole corpus.

Then, every unit not present in the set was decomposed in its phonemes. This task was done by looking in the original labels looking for the bound-

aries between phonemes of the pseudosyllable that have to be split.

After this process of decomposing the least frequent pseudosyllablic units in phonemes we reestimated the frequencies of the 100 chosen units, resulting in that single phonemes were the most frequent units by far, because there were many decomposed units. The frequencies of the phonemes and the pseudosyllables are represented in Figure 4.5. Most of the pseudosyillabes are not frequent. This made the overall process less useful, because we increased the number of classes from 42 to 100 while more than a half of them were rather infrequent. In consequence is not clear that the whole process of using pseudosyllabic units gives us any advantage compared with working only with single phonemes.

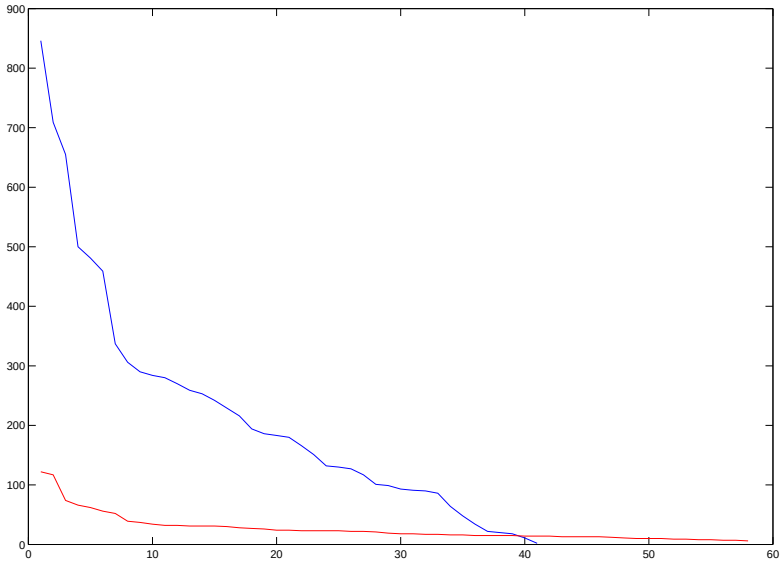

Figure 4.5: Frequencies of the 41 single phonemes (in blue) and the joined phonemes forming 58 pseudosyillabes (in red) in the reduced set of 100 classes (silence is not shown) evaluated in a 1/27 of the corpus.

Anyway, even when the objective of finding a reasonable number of pseudosyllabic units obtained from their representation with the GSDM that could express the German language was not successful, the rules applied to discard bad labeled utterances were still very useful to identify corrupted training data and do not use it in the training process.

4.2. Feature Selection

4.2.1. The SAMME feature selection algorithm

Although SAMME has good performance when classifying (see again [23]), we are not primarily concerned in its final classification $C(\mathbf{x})$. Instead of that, we want it to select the best features from a given set.

Both AdaBoost and SAMME can be used for feature selection with some modifications. The key relies in the first two steps of the each iteration. When fitting a classifier $T^{(m)}(\mathbf{x})$ to the training data, we have freedom to choose the weak learner. If we have a set of $q = 1 \dots Q$ features $\mathbf{x}_q$, each one with N observations, and a weight w_n for each observation, we can fit a classifier $S_q^{(m)}(\mathbf{x_q})$ to each one of the q features, and then take the best $S_q^{(m)}(\mathbf{x_q})$ as $T^{(m)}(\mathbf{x})$.

With this change, the second step of the SAMME algorithm becomes computing for $m = 1$ to M:

1. For $q = 1$ to Q:
   a) Fit a classifier $S_q^{(m)}(\mathbf{x_q})$ to the training data of the feature $\mathbf{x_q}$ using weights w_i.
   b) Compute
   $$err_q^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}(c_i \neq T^{(m)}(\mathbf{x_{qi}}))/\sum_{i=1}^{n} w_i$$
2. Set
   $$err^{(m)} \leftarrow \arg\min_q err_q^{(m)}$$
   and
   $$T^{(m)}(\mathbf{x}) \leftarrow S_q^{(m)}(\mathbf{x_q})$$
   for q such that minimizes $err_q^{(m)}$
3. Compute
   $$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} + \log(K - 1)$$
4. Set
   $$w_i \leftarrow w_i exp(\alpha^{(m)} \mathbb{I}(c_i \neq T^{(m)}(\mathbf{x_i}))), i = 1, 2, \dots, n$$
5. Re-normalize w_i

Choosing $T^{(m)}$ in this way, SAMME will choose at each iteration the feature $\mathbf{x}_q$ that can help it better to achieve a good overall classification. Moreover, it will assign to the feature a weight $\alpha(m)$, that contains information about its importance in the set of features.

4.2.2. Weak learners

We have almost complete freedom to choose $S_q^{(m)}$, being the only requirement that the best $err_q^{(m)}$ is less than $1/K$. As we wanted the time requirements for fitting each classifier to be as smal as possible, we chose a very simple classifier.

The weak classification process was performed by dividing the range $0 \dots 1$ that $\mathbf{x}_q$ can take in r steps. Each step S_r is initialized with an associated value of zero for each class. That value, at the end will be the sum of the w_q of the $\mathbf{x}_q$ that are in that segment for each class. That is, for each q, we examine the value of $\mathbf{x}_q$, and we find the step to which that value belongs. Then, we add its associated weight w_q to the value associated to that step for the class k of $\mathbf{x}_q$. After computing all Q values, we find for each segment the class k with the maximum value. The final classifier $S_q^{(m)}(\mathbf{x}_q)$, will classify according to these value for each interval. This process is depicted in Figure 4.6

This is, in fact, a rough estimation of the probability density particularly prone to overfitting and a very inaccurate classifier, but very fast and simple. More complex classifiers can be used as weak learners, at the cost of increasing the computational time, because they are trained for every Gabor filter on every iteration. Being G the number of Gabor filters and M the number of iterations, if the computational time required to train a weak learner is multiplied by a factor c, then the overall computational time is increased by a factor of cGM. Thus, the weak learners should be as simple as possible if they are expected to select among large numbers of Gabor filters. And, after all, using a number of simple classifiers to build a strong classifier is the idea behind boosting algorithms. Being coherent with this idea, the complexity should be in the side of the strong classifier.

The number r of steps can be adjusted. In the practice, bigger values of r make the final classifier more dependent on the training data. This can be explained by the overfitting of the weak learner.

This is a simple type of stump learner. Stump learners are often used with AdaBoost, because they are simple, they can be calculated quickly, and they are better than guessing (they become guessing for the trivial case, in which the number of steps is 1).

4.2.3. Preselection of Gabor filters

SAMME can perform a selection of the best Gabor filters for classifying. However, it needs an initial selection of Gabor filters, as the potential number is infinite.

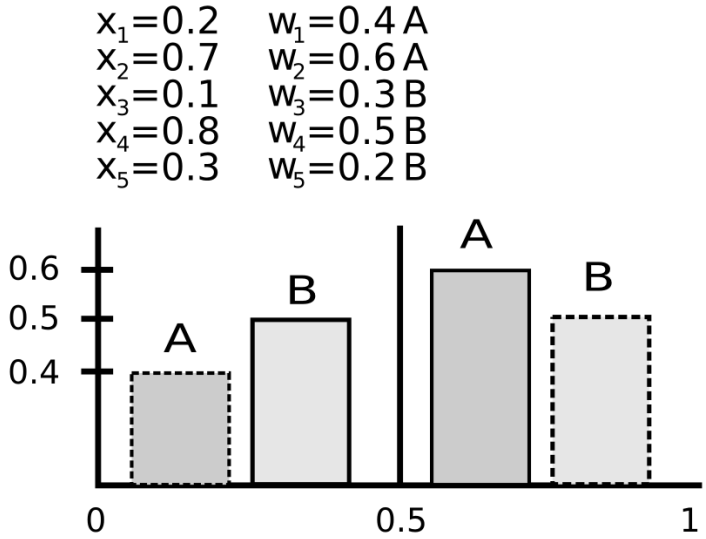

Figure 4.6: Example of a weak classifier with two steps. Values $\mathbf{x}_1$, $\mathbf{x}_3$ and $\mathbf{x}_4$ are in the range of the first step. 3 and 5 have the same class, so we sum their weights and assign the result to the weight of that class in that step. The result is that is more likely that values in the first step belong to class B than to class A. In the second step, the result is different. There is only a sample per class, but the value $\mathbf{x}_2$ with class A has a bigger weight. In consequence, we estimate that is more likely for values in the second step to belong to class A. The weak classifier will classify with that simple rule: Values between 0 and 0.5 will be assigned to class B, and bigger values to class A.

Although other approaches start from random Gabor filters and drop iteratively the worst to arrive to a selection of the Best Gabor filters to classify using Feature Finding Neural Networks, as was done by Kleinschmidt in [10], we chose to employ the SAMME algorithm because AdaBoost has been used successfully in feature extraction for image recognition, and because it has a set of benefits like absence of overfitting, as was discussed before. However, even in the case of random selection of filters, some limits have to be imposed to the size of the Gabor filters, because the space of all the possible Gabor filters is infinite.

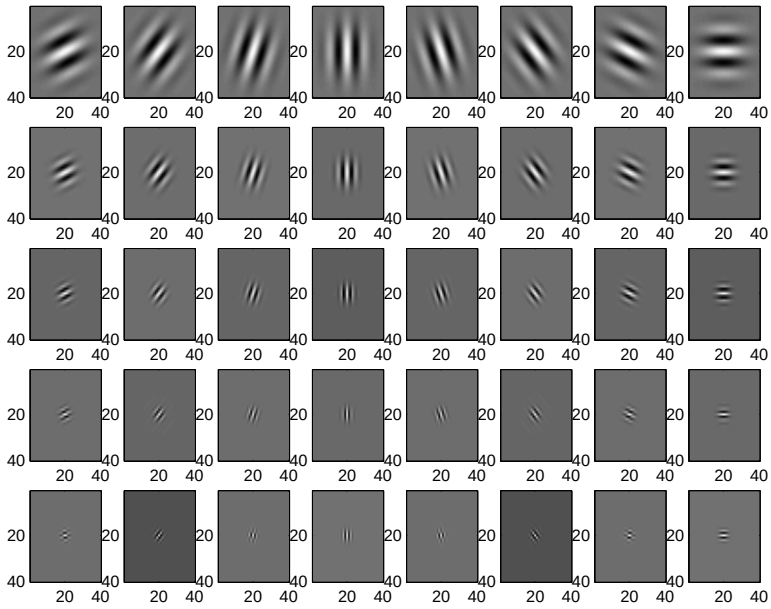

Figure 4.7: Original set of Gabor filters with frequency ranging between 1/10 cycles per point (top) and 1/2 cycles per point (bottom) cycles/point and 8 orientations.

For performing the selection of the eligible Gabor filters, we started with a wide range of sizes varying from $\frac{1}{10}$ cycles per point to $\frac{1}{2}$ cycles per point and 8 orientations, as shown in Figure 4.7. This set is similar to the set used, for instance, by Chengjun and Wechsler for face recognition[12].

However, in our first experiments (see Figure 2.7) it was clear that the smallest Gabor filters were extracting variable and undesirable features with great variability.

This is due mainly to the nature of the GSDM representation. While other representations show patterns of equally distributed stripes, like in Figure 4.9, these patterns are not present in the GSDM representation. Gabor filters are extracting the components of a given frequency in the two-dimensional (time-frequency) representation. That is, when we talk about

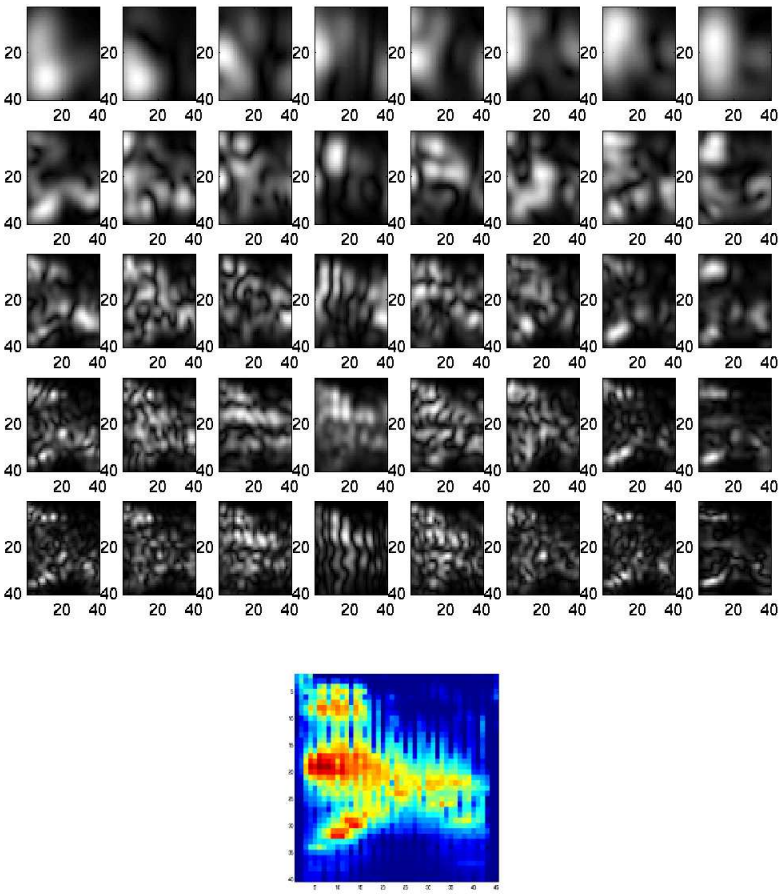

Figure 4.8: Segment /ar/ filtered with 40 Gabor filters with frequency ranging between 1/10 cycles per point (top) and 1/2 cycles per point (bottom) cycles/point and 8 orientations. Smaller Gabor filters present noisy patterns.

the frequency extracted with a Gabor filter, it is not the same frequency as the frequency of the speech. A tone with a given frequency will be represented in the Seneff two dimensional output with an horizontal line. But Gabor filters extract frequencial components in the two dimensional output. If these frequencies in the two dimensional signal are not remarkable, the result is noise.

Even the biggest set of Gabor filters, with $f = \frac{1}{10}$ cycles per point, when evaluated with the same pseudosyllabe, resulted in values very disperse from the mean. Kleinschmidt used Gabor filters with spectrograms that contained two-dimensional patterns that were easier to extract by Gabor filters, but in the GSDMM representation these desired spectro-temporal frequencies do not appear, instead we have cloudy patterns, and thus the zones that get the maximum values when filtered were the ones corresponding to steep changes in the energy of the signal in the selected direction. As abrupt changes are, in the spectral domain, composed by a wide band of frequencies, these frequencies were the ones extracted by the Gabor filters. But abrupt changes, corresponding mostly to the outline of the GSDM signal, are not very stable among observations of the same class, as was appreciated by visual inspection of the resulting filtered samples and comparing them with the mean value.

Then it was decided to use bigger Gabor filters. The new range of filters, with 5 different sizes, had frequencies ranging from $\frac{3}{50}$ cycles per point to $\frac{7}{50}$ cycles per point. This contained the bigger filters of the previous set. The biggest Gabor filters in the set are bigger than the signal. They are stable, since they are so big that extract only information about the general shape of the signal.

Gabor filters are band-pass filters that extract components of a given frequency with one particular orientation in a given location from a two-dimensional representation. The question of whether the information present at those frequencies is useful (stable among observations from the same class and different among observation of different classes) or not, depends on the nature of the original two-dimensional representation. In Figure 4.9 is shown an example of the representation employed by M. Kleinschmidt and B. Meyer in some of their works (see [15], for instance); that is the kind of pattern that small Gabor filters can extract easily. Other applications of Gabor filters, mainly in the image recognition field, use Gabor filters for extracting similar kinds of patterns, such as fingerprints or facial patterns as in Figure 4.10.

But as these spectro-temporal frequencies were not present in the GSDM representation, small-sized Gabor filters, associated with the extraction of high frequency components, they were not useful, because in our cloudy patterns these high frequencies, are not representative of the signal. In Figure 4.3 it can be appreciated that small-sized gabor filters give as a

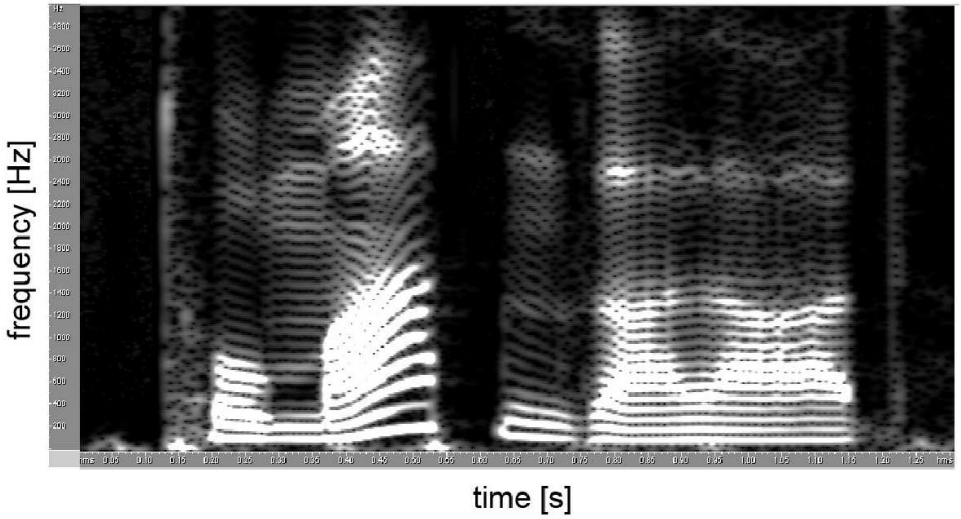

Figure 4.9: Spectrogram of the utterance *tomatensalat* with clear spectro-temporal patterns. From [15].

Figure 4.10: Gabor in face recognition. The pictures on the left show the first 5 selected Gabor filters in [20]. The two rightmost pictures show the position of 200 Gabor filters selected with AdaBoost.

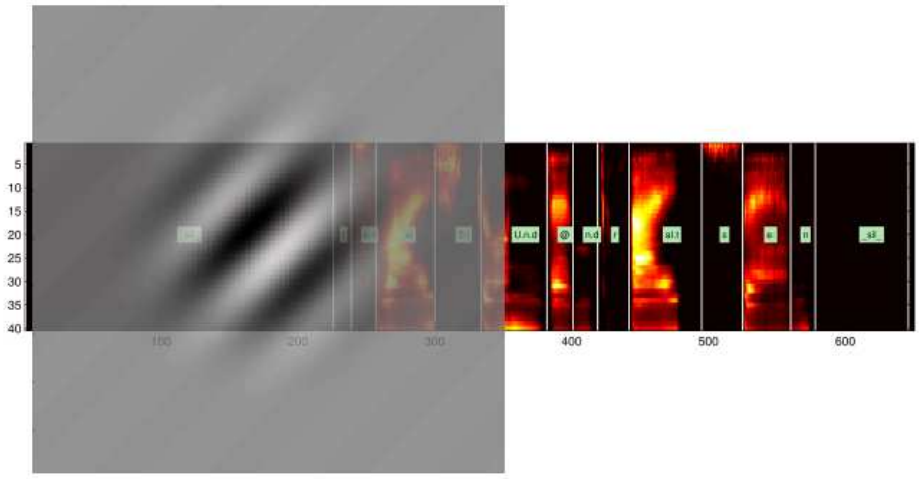

Figure 4.11: Size comparison between an upwards oriented Gabor filter of $f = 3/50$ cycles per point and the 40 channels GSDM signal.

result a noisy pattern, that is not stable. This could be appreciated by visual inspection and performing the selection of SAMME. As it will be shown later, SAMME did not give importance to small Gabor filters (that extract high two-dimensional frequencies) to build a classifier.

On the other hand, big Gabor filters (that extract low two-dimensional frequencies) have two associated problems:

- As they do not extract detail, small zones in the representation are masked with their neighbours.
- Big sized Gabor filters are strongly correlated with filters of other locations in the two-dimensional representation, and even with filters that have different orientation.

Both effects are shown in Figure 4.12. The correlation effect is bad because it limits the number of different Gabor filters that can be used to extract information of the given data, and because it breaks the assumption of different features having diagonal covariance, as is usually assumed by the next stage of the system, the Gaussian Mixture Models.

The amount of information carried by a Gabor filter, given others, was evaluated by calculating the joint entropy of different results of the Gabor filters.

In order to calculate the joint entropy we filtered a large collection of samples with all the filters. For each pair of filters a bidimensional histogram with the results of both filters was constructed. This histogram was divided by the sum of all the values to obtain an estimation of the joint probability distribution. Finally, the joint entropy of two filters is calculated with the formula

$$H(X,Y) = -\sum_{x,y} p_{x,y} \log_2(p_{x,y})$$

This result shows that bigger Gabor filters carry, as expected, more mutual information than the smaller ones. It can be appreciated in Figure 4.13 and Figure 4.14, that takes not in account the position of the filters, that filters of lower frequencies have less joint entropy with the rest, and thus share more mutual information.

4.2.4. Feature selection with SAMME

In practice, there are constraints when dealing with large training sets. As the number of classes increases, the number of training elements should increase as well. Otherwise, if there are not enough samples to learn from,

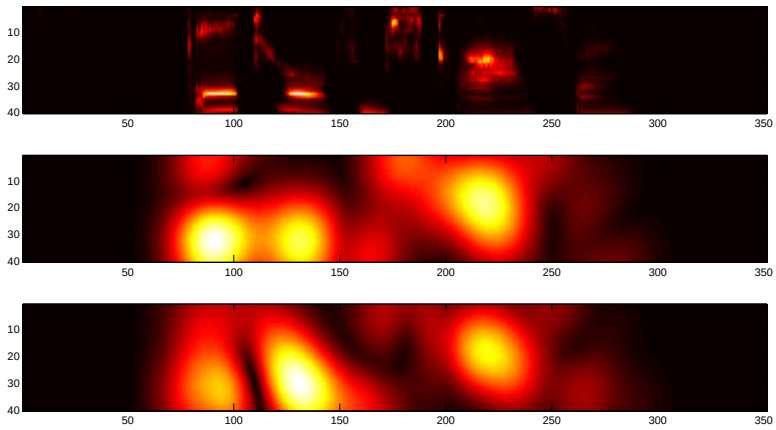

Figure 4.12: Utterance *Beethoven Strasse* filtered with two low frequency filters. From top to bottom, original GSDM representation, utterance filtered with a low frequency filter oriented horizontally, and utterance filtered with a low frequency filter oriented downwards.

the system can learn patterns that are dependent on the particular realizations of the training samples. Ideally, it should learn the patterns that are stable in large realizations of the event to classify.

On the other hand, large training sets have technology requirements in terms of computational speed and memory that sometimes are difficult to fulfill.

SAMME evaluates at each iteration (step a of the algorithm) all the classifiers. In our specific problem, we need all the pseudosyllables to be computed by all the Gabor filters in every position. Each sample is a matrix of 40 frequency channels (the number of channel of the Senef's auditory model) per 70 temporal frames and for a number of 40 different Gabor filters (8 frequency orientations and 5 different sizes).

Each Gabor filter is stored by a matrix of 64 rows and 64 columns of float numbers. In the worst case, using a filter that totally overlaps with the original image, since our samples have 40 rows, when convolving the Gabor filter, 40x64 multiplications of float numbers need to be computed. If we compute the result of each Gabor filter with all the data on every iteration to adjust the weak learners, as was done for instance by Norman Casagrande with Haar features in a two-class problem[1], we would had to restrict severely the size of the training database to get results in a reasonable amount of time. Moreover, we would be doing the same operations on the same data over and over.

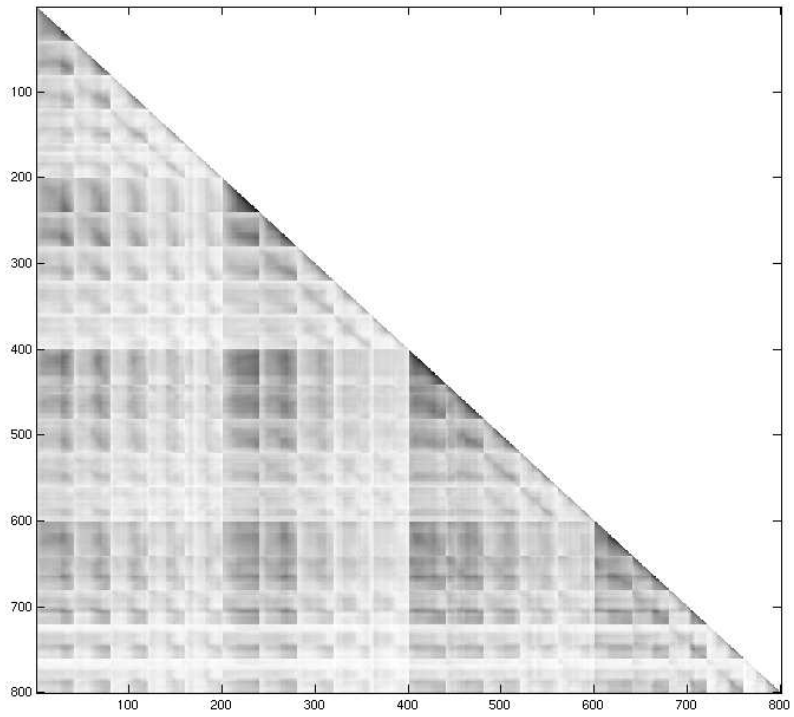

Figure 4.13: Joint entropy between outputs of different Gabor filters from the 20 filters base (5 sizes/frequencies with 4 orientations, here we only considered orientations rotated 45 degrees instead of 22.5 degrees) set in 40 positions (leading to 800 Gabor filters). Black values stand for low joint entropy. First 200 filters correspond to Gabor filters capturing horizontal ripples. Filters in positions 200-399 filter upwards diagonal ripples. Filters in positions 400-699 capture vertical ripples. Finally, last 200 filters correspond to downwards diagonal ripples. In each group, first filters are the biggest. Bigger Gabor filters carry more mutual information.

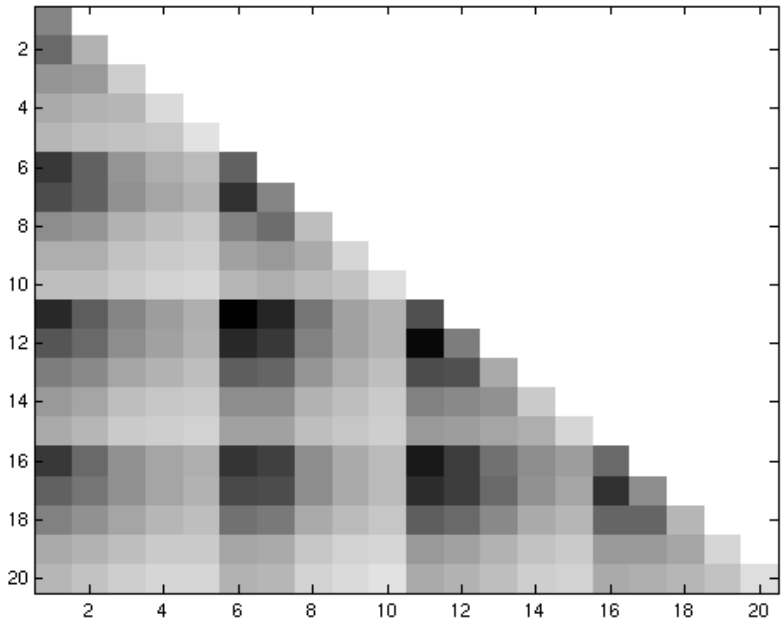

Figure 4.14: Sum of the joint entropy for each filter in its 40 positions. There is a total of 20 filters. Filters in positions 1, 5, 11 and 16 are the biggest ones, while filters in positions 5, 10, 15 and 20 are the smallest ones.

One of the solutions would have been to compute the mean of the value for the filter for all the time frames, and that would have reduced the operations by a factor of 70. This was tried in a posterior step, and the performance was significantly worse.

On the other hand, we can filter the training database in advance, and feed SAMME with the calculated Gabor features. This has the advantage of doing the filtering only once. That is what we did.

While the speed problem had been solved, there still remained the memory problem, as we had to store the result of each utterance filtered with each filter.

For the reasons previously explained, we used big sized Gabor filters, that are strongly correlated. As SAMME makes no distinction between weak learners by discarding the ones with high correlation, it can, and in practice does, select two or more Gabor filters highly correlated. This does not only increase the correlation among features, but, moreover, increases the number of features selected.

Thus, we decided to prune the number of Gabor filters of the preselection in two main ways:

- First, we decided to use Gabor filters with 4 possible orientations, with angles of 0, 45, 90 and 135 degrees, instead of 8 with steps of 22.5 degrees, after observing that the results of discarding that half of the orientations was not significantly worse (about 1% of increment in the training error).

- Second, Gabor filters were evaluated centred in each of the 40 channels of the GSDM and at every temporal frame (from a total of 70). We decided to divide per 16 the number of different locations of the Gabor filters, evaluating in positions $2, 6, 10, \ldots$ both in temporal and spectral axis. Again, the performance was not affected by this reduction of the available number of Gabor filters.

After the pruning, the size of the database was reduced by a factor of 32, while the results did not get much worse, and the resulting selection decreased its correlation. Basically, we discarded very similar Gabor filters given our task. The final set of 20 Gabor filters is shown in Figure 4.15. Of course, these filters can be placed in different locations, so there are still many possibilities and the need of using SAMME.

The number of steps per weak learner was adjusted experimentally. While high number of steps (40) produced the training error to decay quickly (see Figure 4.16), test error does not decrease below 85%.

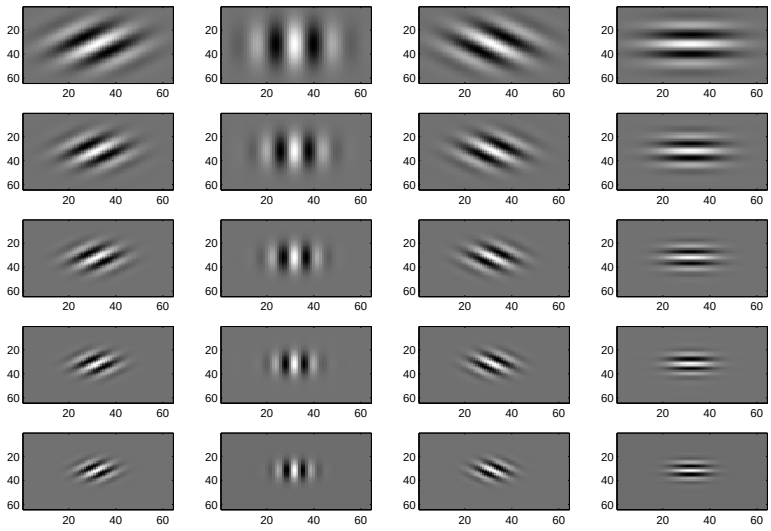

Figure 4.15: Set of 20 Gabor filters used in feature selection

Lower values for the number of steps (10) produced slightly better results, arriving to 82% of test error when evaluated in a 100 class problem (see Figure 4.17.

The selection process was conducted also with single phonemes, discarding finally pseudosylables. Doing that, with 10 branches and 42 classes, the test error decreased to 70%.

Although the test error in SAMME has not a direct influence on the automatic speech recognition performance, because SAMME is not used for classifying, but only for feature selection, these results give an idea about how difficult is the classification task and are useful for tuning the parameters. Although the test error rate was high (70% was the best result), we expected that a complete ASR system using HMM and language models could reduce this error substantially.

4.2.5. Analysis of the weights

SAMME gives a weight to each selected classifier. These weights were accumulated for Gabor filters located in the same spectral location but in different temporal locations, because in the ASR system will be evaluated on every temporal frame. After this accumulation, we obtained a set of weights associated to the Gabor filters.

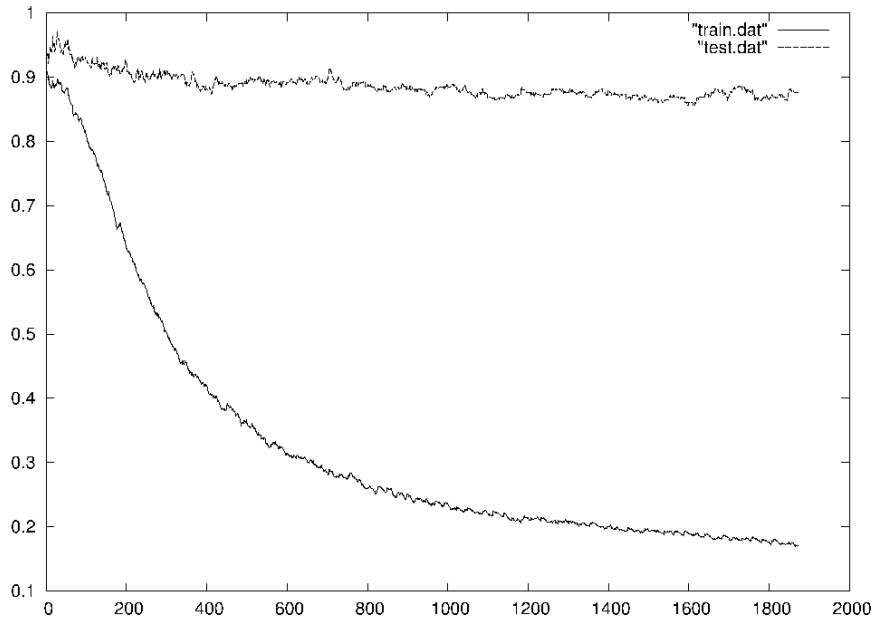

Figure 4.16: Training and test error evolution with 40 branches per weak learner and 100 classes. Horizontal axis stands for the number of iteration.

Utterance 4.18 is filtered with 800 Gabor filters (4 orientations, 5 frequencies, 40 positions) in Figure 4.19 with the results of the 800 Gabor filters. These figures show how Gabor filters extract some of the components of the representation. The problems already mentioned (high correlation, detail masking, recognition of borders instead of frequencial patterns) are visible too.

As can be appreciated in figure 4.20, except for the spectral oriented filters, SAMME selected primary filters of the lowest frequencies (filters in ranges 0-40, 200-240, 400-440). This was commented before: high frequency filters do not extract much information that could be used for classify (except for the pure spectral Gabor filters).

High weights given to pure spectral Gabor filters are related to the importance of pure spectral analysis in speech recognition.

Although we expected the selection of Gabor filters to give more importance to spectro-temporal feature extraction (as it is the purpose of using Gabor filters), experimental results contradicted those expectations.

From this set of weights, the final selection can be done simply by choosing the Gabor filters with higher associated weights.

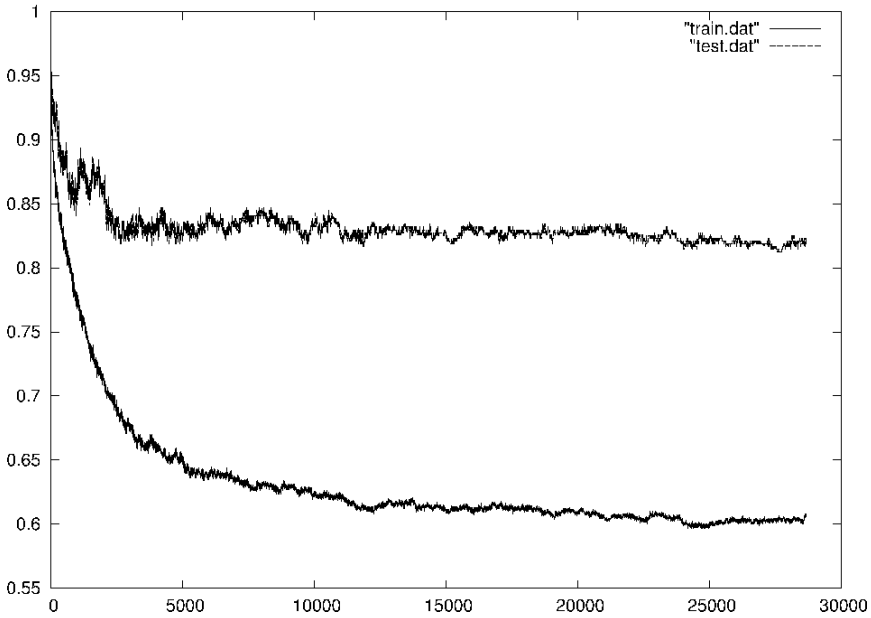

Figure 4.17: Training and test error evolution with 10 branches per weak learner and 100 classes. Horizontal axis stands for the number of iteration.

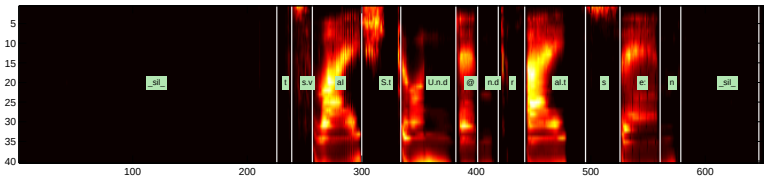

Figure 4.18: GSDM representation of the utterance *zwei Stunden dreizehn.*

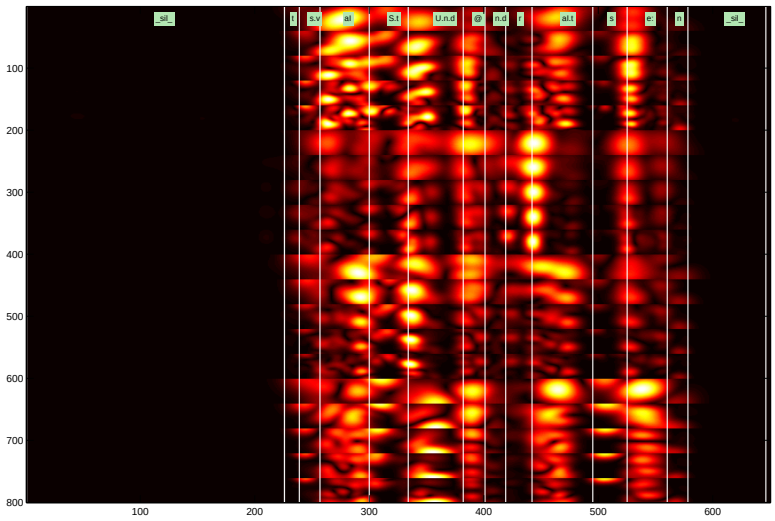

Figure 4.19: Utterance *zwei Stunden dreizehn* (which GSDM representation is provided in the previous Figure) filtered with 20 Gabor filters in 40 different height positions. First five have upwards orientation, 6-10 have pure temporal orientation, 11-15 have downward orientation and 16-20 pure spectral orientation. Inside each set they are ordered from the biggest ($f = 3/50$ cycles pp) to the smallest ($f = 3/50$ cycles pp). On a side note, in this example can be appreciated how pure spectral orientation filters are activated by phonemes /s/ (frames around 320 and 510) and /n/ (frames around 370) and pure temporal filters are activated by steep temporal changes in the energy.

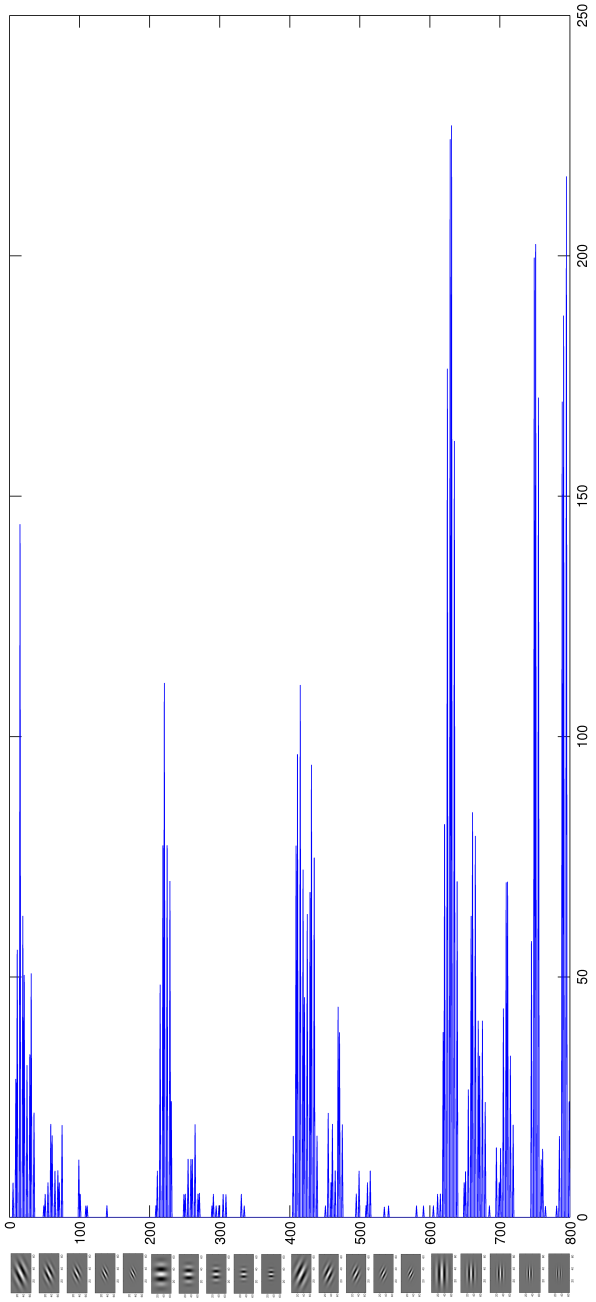

Figure 4.20: Weights for each Gabor filter obtained with SAMME. There are 20 Gabor filters evaluated in 40 spectral positions giving a total of 800 filters. Because of the pruning, 3 every 4 positions were discarded in the pruning and its weight was forced to be zero. First 200 positions are associated with Gabor filters oriented upwards; from 200 to 399 are oriented vertically; from 400 to 599 downwards and from 600 to 799 are oriented horizontally. In each set of 200 filters, they are ordered by size, corresponding the filters in positions 1-39 to the lowest frequency and 160-199 the highest frequency.

4.3. Statistical properties of the new features

The studied features differ drastically from MFCC features in their statistical properties. This difference is crucial, because we are using a typical approach to deal with MFCC features (GMM + HMM) that may not be suitable for this task.

Figure 4.21 shows different histograms of evaluations of the Gabor filters among several utterances of the wovel /i/. They vary between 0 and 1, because the signals were normalized before the training. Taking these histograms as estimators of the probability density function, we can observe that their distribution is far from being Gaussian, as opposite to MFCC. It is formed by a peak and a long tail in the right side of the histogram.

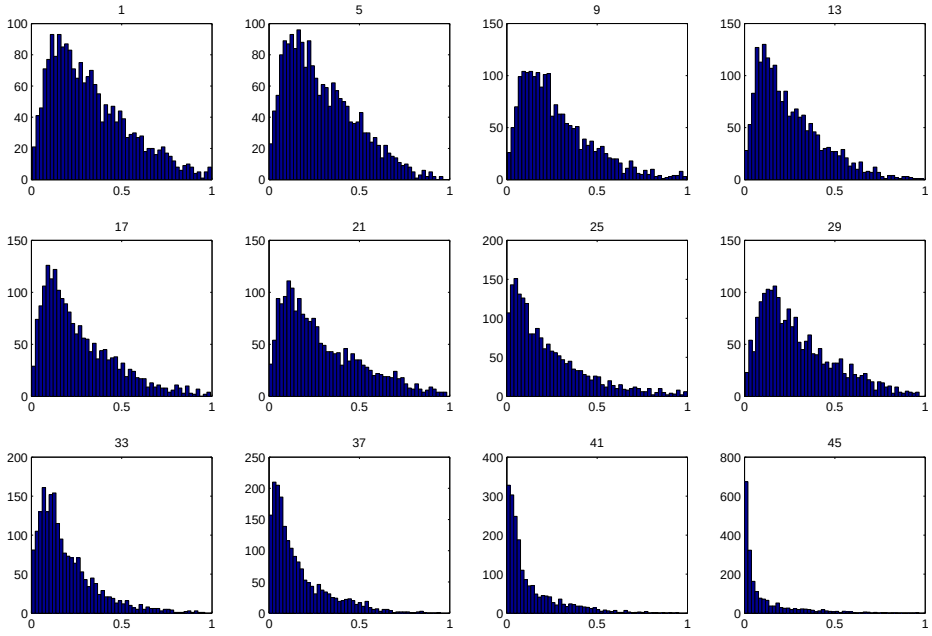

Figure 4.21: Histogram of results of different Gabor filters (their index is on the top of each plot) for the phoneme /i/.

In Figure 4.22 are shown the mean values for each Gabor filter when evaluated on several realizations of different phonemes. As can be observed, different phonemes have different means. Even close phonemes, as /m/ and /n/, show different patterns in some Gabor filters when evaluated from this point of view.

From the point of view of the covariance among Gabor filters, the re-

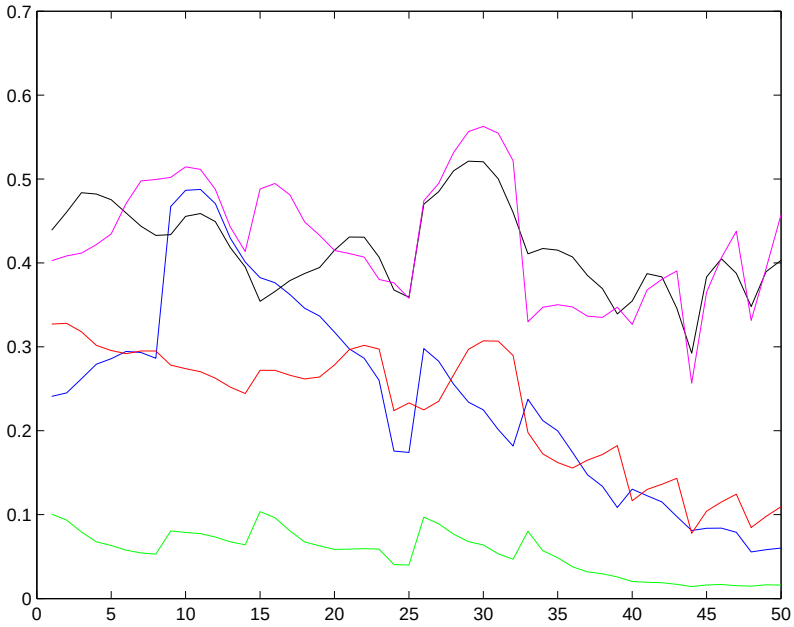

Figure 4.22: Averages of the output for each filter in different phonemes. /a/ in blue, /i/ in red, /z/ in green, /n/ in black and /m/ in magenta. They are the best 50 filters according to SAMME.

sulting covariance matrix, that can be seen at Figure 4.23, is very far from being diagonal, an approximation that can be used in MFCC. This is a known problem when using Gabor filters, and was solved by Kleinschmidt by inserting a multi-layer perceptron (MLP) before the GMM.

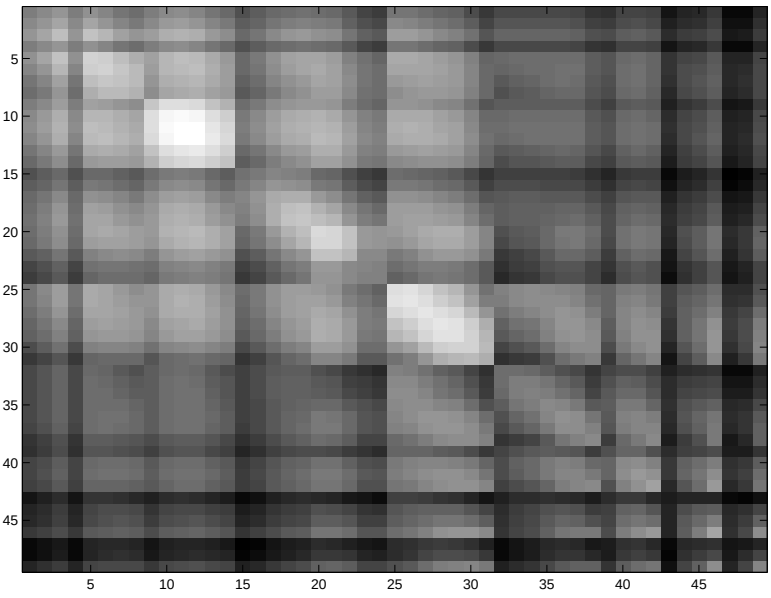

Figure 4.23: Covariance matrix of the 50 Gabor filters evaluated over 100 utterances.

Unfortunately, this approximation was hard-coded in the Janus Speech Recognition Toolkit, and would be difficult to change it in a short term. The employment of a MLP or other technique (LDA or PCA) to decorrelate the features is left to future work on these features, as we had no time to experiment with them.

4.4. ASR results

The framework employed in the complete ASR system was the JANUS Speech Recognizer, initially developed by the Carnegie Mellon University, after that by the University of Kalsruhe and finally by Sony itself. It is a toolkit that allows running TCL scripts to write the modules needed in an ASR system.

At the final testing stage, more problems arose. First, the ASR system available in the laboratory, adapted to this concrete problem and corpus,

had not been tested for two years. This system had two uses in this project: it was the state of the art system to compare our work with, and was the basis to be adapted to the changes analysed in our work. Unfortunately, we discovered that it was not working as expected and had to be debugged, because some changes in the framework had altered the system and it performed very poorly. In consequence, it had to be debugged for two months, time during which some of the problems that appeared were fixed. The corpus had been also partially resampled, in such a way that the labels were wrong in a third of the utterances, and had to be fixed. After the debugging stage, it achieved a best result of 5% word error rate when recognizing words and short sentences of our corpora, as it was described in chapter 3. We trained the system with data from 25 speakers and tested it with the remaining 75. This system had a better performance in the past, so some bugs must still be present and more time would be necessary to find them.

The modified system with Gabor filtered Seneff features, on the other hand, never achieved acceptable error rates, being the best one around 90% word error rate. Even worse, these results are not meaningful, because the JANUS ASR system has problems when dealing with the new features. Assumptions that worked with the MFCC approach were wrong in the modified case. After many tests, we could identify two major problems:

- The covariance matrix of the features was not diagonal, as discussed above.
- In some frames, the ASR system computed a probability of being silence bigger than one (sometimes higher than 300). We did not achieve to track this bug, although we guessed that was related to the fact that the silence usually had a very peaky probability distribution, as opposed to MFCC. That would explain why the bug did not appeared when working with MFCC features, whose probability distribution is not peaky.

But it was impossible to distinguish clearly to what extent these poor results were due to failures of the existing ASR framework or to weakness in the Seneff/Gabor approach studied. So, to evaluate the utility of the GSDM/Gabor approach, may be more interesting the analysis of the properties of these features than this word error rate.

4.5. Conclusions

Although the GSDM representation shows clear distinct patterns for each phoneme and is more robust against noise than MFCC, there still

remains the question of how to post-process these features to be usable by an automatic speech recognition system.

In this work, some new techniques were examined. The final goal was to evaluate also their combination in a full ASR system. Although we could not achieve this final objective, we could learn from the problems that appeared in each subsystem. The experiments suggest that these problems should be explored in depth and solved before concatenating the modules, to show more clearly where the troubles are, separating them from the errors carried from previous stages.

The approach studied of postprocessing with Gabor filters, although applied successfully in the past to other features that show clear spectro-temporal patterns, do not seem to be so promising in combination with the GSDM representation. GSDM has a different kind of pattern, similar to clouds, that makes the usage of high frequency Gabor filters not so useful.

As was shown in the section 4.2, SAMME did almost only select pure spectral and big size Gabor filters. This can be considered an indicator of the convenience of using another kind of post-processing capable of extracting patterns from a cloud-like representation as GSDM is.

In any case, there are problems when feeding directly the secondary features to GMM models. Either the GMM employed should not assume diagonal covariance, as the features are strongly correlated, or some stage or stages should be inserted before the GMM stage. M. Kleinschmimdt and B. Meyer successfully employed the Tandem approach, which consists in using a multi-layer perceptron to feed the GMM[15]. This kind of non-linear classifier is used to provide a set of probabilities for each phoneme as the features to the GMM.

Linear Discriminant Analyisis (LDA) and Principal Component Analysis (PCA) can be also tried for this purpose, although Kleinschmidt, Meyer and Gelbart pointed out that these techniques failed when used with Gabor features[9].

Finally, the work with a new segmentation based on pseudosyllables has to overcome some problems, at least when dealing with previously labeled data. Some errors in the labeling would need a close look, and the application of complex rules (for instance, to deal with plosive phonemes). A more costly approach would be to work with manually labeled data, which would clarify the advantages and drawbacks of the pseudosyllabic units.

Bibliography

[1] Casagrande, N. (Oct. 2005), 'Automatic music classification using boosting algorithms and auditory features', Computer and operational research Department, University of Montreal, Montreal, Master Thesis.

[2] Cosi, P. 'Seneff's Joint Synchrony/Mean Rate Auditory Speech Processing' *http://www.pd.istc.cnr.it/pages/asr-seneff.htm*

[3] Delgutte, B. and Kiang, N. Y. (1984), 'Speech coding in the auditory nerve', *J. Acoust. Soc. Amer.*, 866-919.

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

[5] Giron, F. (2006), 'Correlation analysis for the derivation of speech recognition features based on an auditory model', *TC-STAR Workshop on Speech-to-Speech Translation.*

[6] Hermansky, H. (1998), 'Should recognizers have ears?', *Speech Communication* **25**, 324.

[7] Kearns, M. and Valiant, L.G. (Jan. 1994), 'Cryptographic limitations on learning Boolean formulae and finite automata', *Journal of the Association for Computing Machinery*, **41**(1):6795.

[8] Kearns, M. and Valiant, L. G. (Aug. 1988), 'Learning Boolean formulae or finite automata is as hard as factoring', *Technical Report TR-14-88*, Harvard University Aiken Computation Laboratory.

[9] Kleinschmidt, M., Meyer, B. and Gelbart, D., 'Gabor Feature Extraction for Automatic Speech Recognition', *http://www.icsi.berkeley.edu/Speech/papers/icslp02-gabor/* .

[10] Kleinschmidt, M. and Gelbart, D. (2002), 'Improving word accuracy with Gabor feature extraction', *Proc. ICSLP.*

[11] Lippmann, R. P. (1990), 'Speech recognition by machines and humans', *Speech Communication* **9** 3540.

[12] Liu, C. and Wechsler, H. (2002), 'Gabor feature based classification using the enhanced fisher lineardiscriminant model for face recognition', *IEEE Transactions on Image Processing* **11**, Issue: 4.

[13] Matas, J. and Sochman, J. (Oct. 2004), 'AdaBoost', *Technical report*, Centre for Machine Perception, Czech Technical University, Prague.

[14] Mesaros, J. A. A. (2005), 'The mel-frequency cepstral coefficients in the context of singer indentification', *Technical report*, Queen Mary, University of London.

[15] Meyer, B. (2004), 'Robust speech recognition based on spectro-temporal features', Universitaet Oldenburg, diploma thesis.

[16] B. Meyer, M. Kleinschmidt (2003), 'Robust speech recognition based on localized spectro-temporal features', *Proceedings ESSV*.

[17] Rabiner, L. (1993), 'A tutorial on hidden markov models and selective applications in speech recognition', *Readings in Speech Recognition*, Alex Waibel and K. F. Lee, editors, Morgan Kaufmann Publishers.

[18] Rogina, I. and Waibel, A. (1995) 'The JANUS speech recognizer', *Proc. of the ARPA Spoken Language Systems Technology Workshop*, Austin, TX.

[19] Seneff, S. (1990), 'A joint synchrony/mean-rate model of auditory speech processing', *J. Phonetics* **16**, 55-76.

[20] Shen, L., Bai, L., Bardsley, D, and Wang, Y. *Lecture notes in computer science* ISSN 0302-9743

[21] Stevens, Volkman and Newman (1937), *J. Acoust. Soc. Am*, **8**(3) 185-190.

[22] Yost, W. A. and Nielsen, D. W., *Fundamentals of hearing - an introduction*, New York: Holt, Richart and Winston.

[23] Zhu, J., Rosset, S., Zou, H. and Hastie, T. (2005) 'Multi-class AdaBoost', *Technical report*, Stanford University.