Implementation of Gaussian Mixture Models in .Net technology for automatic speech recognition

María Álvarez Rodríguez

SUPERVISOR:
Prof. Mariusz Ziołko

Krakow 2011
# Table of contents

1. **INTRODUCTION** .............................................................................................................. 6

2. **PATTERN RECOGNITION** .............................................................................................. 10
   2.1. Definition .................................................................................................................. 10
   2.2. Unsupervised versus supervised classification ...................................................... 12
   2.3. Data Clustering .......................................................................................................... 12
       2.3.1. Mixture Model-based clustering ..................................................................... 15

3. **AUTOMATIC SPEECH RECOGNITION** ................................................................. 17
   3.1. Advantages and difficulties of ASR systems ......................................................... 18
   3.2. Types of recognition .............................................................................................. 19
   3.3. Characterization of an ASR system ...................................................................... 20
   3.4. ASR approaches ...................................................................................................... 21
       3.4.1. Acoustic-phonetic approach ....................................................................... 21
       3.4.2. Pattern recognition approach ..................................................................... 22
       3.4.3. Artificial Intelligence approach .................................................................. 22
   3.5. Structure of a typical speech recognition system ................................................. 22
   3.6. Applications of ASR systems ................................................................................. 26

4. **GAUSSIAN MIXTURE MODELS** ................................................................................. 28

5. **EXPECTATION MAXIMIZATION ALGORITHM** ....................................................... 33
   5.1. Application to Gaussian Mixture Model ............................................................... 34
       5.1.1. Convergence and initialization .................................................................. 40
       5.1.2. Summarizing the algorithm ....................................................................... 40

6. **GMM CLASSIFIER** ........................................................................................................ 42
   6.1. Development environment ..................................................................................... 42
   6.2. Corpora (Database Description) ............................................................................ 42
   6.3. General description of the tool ............................................................................. 42
       6.3.1. Structure ....................................................................................................... 42
6.3.2. Inputs ........................................................................................................ 43
6.3.3. GMM Classifier .......................................................................................... 43
6.3.4. General working of the program ............................................................... 45

7. EXPERIMENTS AND RESULTS ................................................................... 48

8. CONCLUSIONS ............................................................................................. 49

9. BIBLIOGRAPHY .............................................................................................. 50
1. Introduction

For a long time, the scientific world has been working in the development of systems capable of interchanging spoken information with human beings. With this purpose it have been tried the development of systems which are able to receive spoken orders and/or messages, interpret these messages, execute the given tasks and present results.

Speech is the most often used communication way among human beings so it results natural to apply it for the communication between humans and machines. However, in spite of the simplicity that presents the spoken communication between humans, transferring this process to the interaction with machines is not a simple task at all. It is necessary to face a multitude of difficulties that need to be solved for these systems to work correctly, and in spite of the advances that have been done in this field, a performance with a 100% correction have not been achieved yet. This fact causes that although giving spoken orders to the machines is much faster than using the keyboard, the use of these systems for the medium user is lower than it could be expected.

However, despite the difficulties that have to be faced, advances in the field of speech technologies are day and day more significant. The development of better algorithms and more precise modeling, together with the appearance of informatics systems more powerful and cheaper have made easier the advance in speaker recognition systems in the last few years. The scenery has change from a situation in which systems wasn’t able to recognize more than a small set of words pronounced by a single speaker to another with medium-large vocabularies and systems able to recognize connected words or identify key words in a continuous speech with a considerable independence of the speaker. Nowadays, some different systems have been developed in this field. Among them could be highlighted alarm systems, text translator systems, speaker recognition machines, etc

The aim described for this MCs. Project was the implementation of a part of a speech recognition system, namely, a classifier of voice samples. For doing this task, the .Net platform will be used. This tool will perform the classification through the design of Gaussian Mixture Models (GMM). This models consist of weighted sums of Gaussian density functions whose parameters (which will be \textit{a priori} unknown is this case) will be calculated through the execution of a classification algorithm called Expectation Maximization (EM). Both GMM and EM will be explaining in detail in next sections of this thesis.

In this first part of the thesis, we will present the different technologies that will take part in the development of the mentioned tool. We will start giving a general vision of
the speech technology (field in which the tool is outlined). Then, we will comment briefly some exact technologies that will be used for its development.

1. Speech technology

Speech technology is an investigation field whose principal aim is to build systems capable of interacting with human beings in a spoken way. Inside this area it is possible to difference between a few different fields: Speech synthesis has the aim of developing techniques and algorithms that allow creating systems with the capacity of producing voice.

- Speech recognition and speaker verification include the techniques used respectively, for speaker identification through the analysis of the spoken speech and the authentication of the identity declared by the speaker in the basis of his/her voice analysis.

- Speech compression refers to the technologies used for voice compression with the purpose of storing and playback.

- Automatic speech recognition (ASR) has as specific aim to develop techniques and algorithms which allow creating systems with the capacity of listening and interpreting speaking messages.

The tool designed in this thesis is outlined within the last one of these technologies; therefore this will be the field in which this thesis will focus.

2. General scheme of Speech Recognition Systems

ASR systems usually operate in two stages. The first stage is called training and the other is called recognizing or identification.

During the training phase, it is present to the system a certain amount of speech elements (phonemes, words, phrases, sentences, etc.) which are desired to be memorized by the system is presented. During the recognition phase (once the first stage is concluded), the system is asked to identify a particular pronunciation. It is important to note that the pronunciation to be recognized has not necessary to be one of those used in the training stage.

It is important to point out that the stored information is constituted by properties extracted from the training pronunciations, so what is actually stored is not the set of pronunciations but the properties of that set.
An ideal situation will be that systems respond in real time, however, the main problem of currently used speech recognition systems is that with the aim of dealing with large vocabularies they have to store a lot of data which has as a consequence a low response speed.

3. **Statistic tools used in the development of speech recognition systems**

In every recognition system the statistic has taken part in different ways through the application of different techniques. Two commonly used techniques in this field are Hidden Markov Models (HMM) and clustering techniques.

Hidden Markov Models have been the dominant solution in ASR systems during the last few years. HMM assume that the system under study follows a Markov process with unknown parameters. The main task consists in determining the hidden parameters from the observed parameters.

Other typical tools in the design of ASR systems are the clustering techniques. For identifying every single different sound that a recognition system can manipulate, a training process is firstly necessary; this training consists of a clustering process which groups the observation vectors which can be considered “similar” in the sense of a metric (the most generally used are Euclidian and Itakura-Saito distances). Each group is associated to a different sound and it will be so many groups as sounds the recognition system can handle.

Within the diverse techniques of clustering we will focus on a type called mixture model-based clustering, more precisely in Gaussian Mixture Models.

4. **GMM Classifier**

The implemented tool consists of a classifier of voice samples based in mixture models which will identify the input data using maximum likelihood criteria.

It can be seen as a classification problem in which we have K classes that correspond with the different phonemes that can be identified. The classes will be represented by the models $\theta_1, \theta_2, ..., \theta_K$, which will be Gaussian Mixture Models. In addition, we have a set of feature vectors $X=\{x_1, ..., x_M\}$ corresponding with the audio input we want to classify. The objective will be find the model which has the biggest a posteriori probability for the input feature vector.

The decision is made by computing the likelihood based on the probability density functions (pdfs) of the feature vector. Parameters that define these density functions have to be estimated *a priori*. 
There are many model structures general enough to characterize a speech density function. Here we focus on the Gaussian mixture model, which will be completely defined in section 4.

Given a sequence of feature vectors, the GMM parameters can be estimated iteratively using the expectation maximization algorithm, which also will be explained further in section 5.

Along the next sections of this thesis the technologies mentioned in this introduction will be described in more detail. We will proceed explaining them from the most general to the most concrete fields. Thus, we will start giving a general vision of pattern recognition to continue with an explanation of the most relevant aspects about speech recognition. We will continue talking about Gaussian Mixture Models and Expectation-Maximization algorithm so these are the specific technologies in which the classifier is based.

Once the theoretical basis has been explained, a detailed explanation of the designed classification tool will be exposed. Finally, in section 7, a series of tests done for observing the behavior of different aspects of the tool will be explained and presented.
2. Pattern Recognition

2.1. Definition

A pattern is defined as the common denominator among the multiple instances of an entity. For example, a typical pattern is the fingerprint pattern which refers to the commonality in all fingerprint images. Another pattern could be the set of repeated colors or textures in an image. Pattern recognition is, as been defined in [22], the science which has the aim of classifying a group of objects into several categories or classes. These objects can have different nature (images, signals...) but all of them constitute a type of measurement that need to be classified. Typically, these objects are referred as patterns.

Patterns are obtained through the processes of segmentation, feature extraction and description where each object is finally described by a collection of descriptors.

The objects to classify are normally described by vectors of features, which together constitute a description of all known characteristics of the object.

The feature vectors whose true class is known and which are used for the design of the classifier are known as training feature vectors.

Pattern recognition encloses sub disciplines like discriminative analysis, feature extraction, error estimation, cluster analysis, grammatical inference and parsing (sometimes called syntactical pattern recognition).

Pattern recognition systems have very diverse applications. Some of the most relevant and currently used are:

- Character recognition: one of the most popular applications of pattern recognition because both printed and handwritten characters are easily recognizable. It is a technique used frequently both in banking and postal applications.

- Speech recognition: the analysis of the spoken words is very important form user interaction with machines and is currently used in many applications. The characteristics and applications of this technology will be treated in detail in the next section.

- Medical applications: many medical tests use pattern recognition for its realization, like detection of irregularities in X-ray images, detection of infected cells, counting blood cells, etc.
- Personal identification systems: very important for security applications in many different environments like airports, shops, etc. Recognition can be based on face, iris or fingerprint.

- Interpretation of photography made from the air or by a satellite. It is very important for cartography, agricultural inspection, etc.

- Object recognition: with important applications for people with visual disability.

Generally, a pattern recognition system follows a modular structure like the one below.

![Pattern recognition system architecture](image)

**Figure 1. Pattern recognition system architecture**

The main processes involved are:

- Data acquisition: usually done using sensors which have to be able to transform physical or chemistry magnitudes into electrical ones.

- Feature extraction: the most important part of the process. It is the process of generating characteristics that represent the input objects in some sense and will be used in the process of data classification. Usually these characteristics are represented in vectors called feature vectors.

- Classification and decision: classification consists of assigning the different feature vectors to groups or classes on the basis of the extracted features. According to the existence or not of a previous set that the system would use for learning, the classification can be supervised or not supervised.
2.2. Unsupervised versus supervised classification

When facing a classification problem we can find two types of situations regarding to the data availability. These two situations define two different types of classification called, respectively, supervised and unsupervised classification.

When the set of training data we have to work with, is available, the classifier is designed by exploiting this *a priori* known information. This is what is known as supervised classification. However, this is not always the case, and we can find that part of the data is not available. In this type of problem, we are given a data set and the goal is to find the underlying similarities and group “similar” data together. This is known as unsupervised classification.

From a theoretical point of view, supervised and unsupervised learning differ only in the causal structure of the model. In supervised learning, the machine has the a group of inputs and also is given another set of observations composed by a sequence of desired outputs. With the knowledge of these two groups the machine has to learn to produce the correct output given a new input. So, in this type of learning process the inputs are assumed to be at the beginning and outputs at the end of the causal chain.

On the other hand, in unsupervised learning, there is no explicit target outputs, all the observations are assumed to be caused by latent variables, that is, the observations are assumed to be at the end of the causal chain.

2.3. Data Clustering

Between the most important methods of unsupervised learning we can find the technique called data clustering or simply, clustering. It consists of the assignment of a set of observation into a group of different subsets, called clusters, so all the observation assigned to the same cluster are similar in some sense. This similarity can be measured in terms of the distance, the statistical properties of the data, etc (commonly the definition of similarity is the distance function). So, the criteria used for the classification will be dependent on the final aim of the clustering and it will be supplied by the user in such a way that the result of the clustering will suit their needs. For doing this classification, clustering techniques doesn’t use any prior class information. This is the reason why clustering is considered an unsupervised classification method.

Clustering is a common technique for statistical data analysis and it is used in many fields like machine learning, data mining, image analysis or pattern recognition.
A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some way”. A cluster is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.

The picture below [1], shows graphically a simple example of clustering. In this case, the criteria followed for the classification has been the distance among the input data. Following this criteria (the most tipical one in clustering tasks) the different data result classified in the four groups (clusters) that can be seen in the graphic of the right.

![Figure 2. Example of clustering](image)

One important task in clustering is if it is determined or not, *a priori*, the number of clusters in which the data will be classified. Many clustering algorithms require the specification of the number of clusters to produce in the input data set, prior to execution of the algorithm. In this case it can be a sensitive task because it will determinate the way data will be classified. Sometimes, it will be necessary some training for evaluating the ideal number of clusters. In other cases, this number will be provided for the nature of the classification. This, usually happens when the aim is characterized some type of unknown distribution and is the type of clustering we face in this paper.

According to [22] there are several steps that must be followed in order to developing a clustering task:

- Feature selection: it is important to select the features with two goals in mind: encoding as much information as possible and minimizing the information redundancy among them.
- Proximity measure: measuring of the “amount” of similarity between two feature vectors. It is assumed that all features contribute equally to the computation of the proximity measure.

- Clustering criterion: it is expressed via a cost function or a group of rules which depends on the idea the expert has about what is a “sensible” cluster.

- Clustering algorithms: it is necessary to choose a specific algorithm that unravels the clustering structure of the data set.

- Validation of the results: once the results of the clustering have been obtained it is necessary to make some tests that verify their correctness.

- Interpretation of the results.

The application of cluster analysis are multiple and varied. Some of them are those mentioned in the following list [22]:

- Data reduction: cluster analysis can be used in order to group the data into a “sensible” number of clusters and process each cluster as a single entity, so we handle less large amount of data.

- Hypothesis generation: clustering techniques are sometimes applied to infer some hypotheses concerning the nature of the data.

- Hypothesis testing: clustering analysis can be used also for the verification of the validity of a specific hypothesis.

- Prediction based on groups: in this case we apply cluster analysis to the available data set, and the resulting clusters are characterized based on the characteristics of the patterns by which they are formed. In the sequel, if we are given an unknown pattern. We can determine the cluster to which it is more likely to belong and we characterize it based on the characterization of the respective cluster. This is what happens in the case of GMM analysis.

Clustering algorithms may be classified as listed below:

- Exclusive Clustering: data are grouped in an exclusive way, so that if a certain datum belongs to a definite cluster then it could not be included in another cluster.

- Overlapping Clustering: uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership.
- Hierarchical Clustering: is based on the union between the two nearest clusters. The beginning condition is realized by setting every datum as a cluster. After a few iterations it reaches the final clusters wanted.

- Probabilistic Clustering: use a completely probabilistic approach and is the most commonly used in ASR systems.

We will be interested in the last type, the clustering based on probabilities, which includes the Mixture Model-based clustering which is the one we will focus in this paper.

### 2.3.1. Mixture Model-based clustering

One of the multiple approaching to clustering problems is the model-based clustering, which consists of representing each cluster with a parametric representation like Gaussian or Poisson. The entire data set is modeled by a mixture (a weighted sum) of these distributions. Each individual distribution used for modeling a cluster is normally referred like a component distribution.

One of the most used methods in this type of clustering is the one based in the mixture of Gaussians. In this case, each cluster is considered a Gaussian distribution with a specific mean and variance.

An example similar to the one before can be seen in picture 3 [1]. This time each cluster is represented by a Gaussian and in each Gaussian the mean defines the center and the variance is represented by the grey circle.
Samples will be classified through the execution of some type of algorithm which assigns a cluster for each of them normally based on a criteria of maximizing the probability.
3. Automatic Speech Recognition

Speech recognition, also known as automatic speech recognition or computer speech recognition is a technology framed in the field of pattern recognition, which consists of the process of transforming spoken words, recorded by an electronic device like a microphone or a phone, into text. The recognized words can be the final result of the system as well as the input for a more complex processing. It is important not to confuse speech recognition with speaker recognition since the latter refers to the capability to identify a specific speaker by comparing unknown recorded voices to known voice samples to identify similar and dissimilar characteristics. Speech recognition is a broader solution which refers to the technology that can recognize the words without an assignment to a specific speaker.

The first tries of developing ASR systems come from the 50s. These firsts systems allowed the recognizing of a small vocabulary of isolated words with orders of approximately 10 words spoken by a single speaker. In the 70s the first systems of isolated words recognizing appeared. In this decade the recognizing systems were based on template methods. However, in the 80s the use of these methods decrease in favor of a new tool: the Hidden Markov Models, widely used in current systems. In this decade some approximations based on neuronal networks were also introduced. Today, it is possible to work with large vocabularies and continuous speech and some ASR systems have already been commercialized. Most of these systems are based on HMM or its hybrid version with neuronal networks and normally achieved a 95% of accuracy for a single speaker, with a good quality microphone and in a low-noise environment. In other conditions these systems performance gets worse quickly due to different factors that difficult the recognition problem which will be explain in the following section.
As we will see at the end of this section, in the last few years, the use of ASR systems has reached many and diverse applications. They include, among others, voice dialing, call routing, control of home automation systems or speech-to-text processing.

### 3.1. Advantages and difficulties of ASR systems

The use of voice and more concretely, the speech recognition technology, as a way to give orders to computers, offers a few advantages respect the traditional way of communication between user and machine (i.e. keyboard and mouse).

- It makes the communication faster and more comfortable for users since being the natural communication way for human beings, no especial ability is necessary.

- It allows having hands free for doing any other task while giving orders using the voice.

- It also allows mobility because spoken orders can be told from a certain distance while the use of a keyboard limits much more the user mobility.

- It allows remote access since a network like the telephonic one can be use for accessing to a computer.
Finally, it allows the decreasing of control panels’ dimension. For instance, it is easy to imagine how much manual controls could be suppressed from a plane control panel if the voice could be used as a way to communication.

On the other hand, even though the speaking process is pretty simple for human beings, its computational treatment shows a certain amount of difficulties. The problem of speech recognition presents an interdisciplinary nature and it is necessary to apply techniques and knowledge from different areas [15]: signal processing, physics (acoustics), pattern recognition, communication and information theory, linguistics, physiology, computer science and psychology. In addition, a lot of factors have an influence in the difficulty of the ASR process and therefore in its performance.

- Continuity. Natural language lacks in separators between language units. Sometimes this lack happens even between words.
- Variability of sounds, fundamentally due to the different accents or way of speaking of each speaker.
- Variability in the production of sounds due to the different ways of sound production, coarticulation and inclusion.
- Noise and interferences. Human beings can recognize the speaking in adverse conditions, with a low signal/noise relationship or in presence of other interfering noises thanks to the characteristics of the human auditing system, but this is a much more difficult task for automatic system.
- Acoustic ambiguities. Sometimes it is not possible to map acoustic events to their correspondent phonetic symbols; it is not possible a good codification since the system does not have access to all the knowledge sources that a person uses during a conversation.

3.2. Types of recognition

Assuming the problems exposed in the previous section some restrictions have been imposed to the different methodologies and architectures with the purpose of simplify the problem. The restrictions imposed affect fundamentally to some aspects to the voice signal to recognize, like the number of speakers, the size of the vocabulary or the acceptable variability in the speech signal. According to these restrictions, ASR systems can be classified according to several criteria.

1. According to the speaker. It is possible to distinguish between three types of recognizers: monolocutor systems (the system is training and works only for a single speaker), multilocutor systems (both the training and evaluation
processes are done for the same set of speakers) and speaker independent systems (different set of speakers for training and testing phases).

2. According to the type of speech required for the system we differ between Isolated Word Recognition, Connected Word Recognition, Continuous Speech Recognition and Spontaneous Speech Recognition.

3. According to the possible signal distortions we separate recognizers in two types. Clean speech recognizers are those trained and tested in laboratory conditions, while robust speech recognizers can be used in real environments where factors like noise difficult the recognition task.

3.3. Characterization of an ASR system

ASR systems can be characterized by many parameters. Some of them are shown below [5]:

• Speaking mode

There are two modalities, isolated words or continuous speech. In isolated-word systems the speaker pauses briefly between words, which is not necessary on a continuous speech system. The first of these systems is simpler because isolated words are more easily recognizable as the bounds between them are much clearer.

• Speaking style

Again, we face two alternatives, read speech and spontaneous speech. Read speech is easier to recognize than spontaneous speech because separation between words is more defined and the rhythm and intonation use to be constant.

• Speaker dependency

Speech recognition systems can be speaker dependent or speaker independent. In speaker dependent systems, speakers have to provide a sample of their speech before using the system while speaker independent systems can recognize the speech for a variety of speakers.

• Vocabulary

It can be small or large. Speech recognition is generally easier when the vocabulary is small. This is mainly due to two reasons. Firstly, because with the increase of the number of words is easier the appearance of words similar to each other and secondly, because the processed time grows with the increasing of the number of words to compare. One possible solution to this problem is to recognize linguistic
units smaller than the word (phonemes) because these units have a limited number which is inferior to the number of possible words. However, the difficulty of recognizing these units is bigger due to its duration is very short and the border between two consecutive units is more difficult to establish.

- Language model

There are two main options; finite state or context-sensitive.

- Perplexity

It refers to the number of words that can follow a specific word once the language model has been applied.

- Transducer

The type and location of the microphone also affect to the system performance.

### 3.4. ASR approaches

For the development of ASR systems, some different techniques have been followed. The most general approaches to this task are the three listed below. The three of them follow a different philosophy but in the three cases we can talk about a training phase and another phase consisting of the recognizing itself. Also, in the three approaches the first necessary step is the parameterization of the voice signal in a set of parameters or features suitable to each system.

#### 3.4.1. Acoustic-phonetic approach

This type of approach consists mainly in detecting basic sounds and assigning them concrete labels. This approach basis is the hypothesis that in spoken language a number of different finite phonetic units (phonemes) exist and that these units can be characterized with a set of acoustic properties.

The recognizing process consists in two steps. Firstly, we will do the segmentation and labeled stage. The signal is divided in acoustic regions which one or more phonemes are assigned to. Secondly, the system will determine a valid word (or a set of words) through the labeled phonemes from the first step.
3.4.2. Pattern recognition approach

This approach consists also of two steps. The first step will consist in the pattern training while in the second step the comparison with the patterns will be done.

The main characteristic of this approach is that it uses a mathematical basis that establishes consistent representation for the voice patterns that can be used for the comparison. Voice patterns representation can be a template or a statistic model (HMM or, in our case, GMM) which can be applied to a sound, a word or a phrase. In the comparison step, a direct comparison between the unknown voice signal (the one we desire to recognize) and all the possible patterns learned in the training step will be done.

This last approach is the most used in current applications and it will be the one used in the tool developed in this thesis.

3.4.3. Artificial Intelligence approach

In this approach the aim is trying to automate the recognition procedure according to the way in which human beings apply their intelligence in the visualization, analysis and voice characterization based in a set of acoustic features. The most used techniques in this field are those based on artificial neural networks.

3.5. Structure of a typical speech recognition system

The task of ASR is to take as input an acoustic waveform and produce as output a string of words. The figure below shows the components of a typical speech recognition system.
Let us explain each of these modules in a bit more detail.

- **Speech signal processing**
  
The first operation that a speech recognizer must do is processing the voice signal that constitutes the system input, with the purpose of extracting the relevant acoustic information for the task that has to be done. Typically, this process implies the transformation of the analog speech signal into a digital signal. In this first level it is necessary to note that some perturbations, like noise, can go together with the voice and they should be eliminated or, at least, reduced.

- **Feature extraction**
  
This subsystem is in charge of transforming the speech signal in a set of vectors containing the features which characterized the signal. These vectors are known as feature vectors.

The feature extraction consists in several typical steps that don’t differ much between different systems.

To capture the dynamics of the vocal tract movements, the short-term spectrum is typically computed every 10-20 ms using a window (typically Hamming or Hanning window) of 20-30 msec.
Each one of the frames obtained in the previous step is analyzed and represented by a parameter vector which contains spectral information. Some different techniques for the frame spectral analysis exist; among them can be highlighted Linear Predictive Coding (LPC) analysis, Perceptual Linear Prediction (LPL) analysis and Mel Cepstrum analysis. Here, we will focus on Mel Cepstrum technique since it will be the one used in the designed application.

The cepstrum was defined by as the inverse Fourier transform of the log magnitude spectrum of a signal [15]. The cepstrum has some good properties that make it adequate for using in ASR systems. It allows dimension reduction due to the presence of dominant energy at the first few coefficients. In addition, the low correlation among different coefficients makes it suitable for Gaussian modeling with diagonal covariance matrices.

We call Mel-cepstrum to the cepstrum computed after a non-linear frequency wrapping onto a perceptual frequency scale, the Mel-frequency scale. Since it is an inverse Fourier transform, the resulting coefficients are called Mel Frequency Cepstrum Coefficients (MFCC).

The common way for obtaining MFCCs is the following one:

1. Take the Fourier transform of the previous windowed signal.
2. Map the powers of the spectrum obtained above onto the Mel scale, using triangular overlapping windows.
3. Take the logs of the powers at each of the Mel frequencies.
4. Take the discrete cosine transform of the list of Mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

- Training data

The training data are composed by feature vectors whose class of belonging is known \textit{a priori} and which will be used for determining the model parameters values.

- Acoustic models

In order to analyze the speech frames for their acoustic content, we need a set of acoustic models. There are many kinds of acoustic models, varying in their representation, granularity, context dependence, and other properties. The most popular model representation for acoustic models are template and stochastic representations (shown in figure 7 [20]).
Template representation is the simplest one. It is simply storing a sample of the unit of speech to be modeled, e.g., a recording of a word. Then, the process of recognizing a word would consist in comparing the word given with all the stored templates and select the one with the biggest similarity. However, despite its simple behavior template models are not always recommended mainly due to their incapability of modeling acoustic variability and also the fact that they are limited to systems that recognize whole words as it is hard to record or segment a sample shorter than a word.

Stochastic representation is more flexible than the previous one and for this reason it is used in larger systems. Stochastic models are the most popular approach for ASR. In this approach, words are represented using a probability distribution. This distributions can be modeled parametrically, by assuming that they have a simple shape (like Gaussian density functions) and then trying to find the parameters that describe it; or non-parametrically, by representing the distribution directly (like in neural networks).

- **Modeling**

One of the most important parts in a speech recognition system is the stage in which the model construction is done. The modeling subsystem will identify the different sounds presents in the pronunciation. For doing this task, it uses each vector of the feature vectors sequence obtained in the previous module. A number of procedures have been developed for acoustic modeling. In this case, as mentioned before, we will model our data using a stochastic model, the GMM.
The previous measures will be then used for searching the most probably word, applying the restrictions imposed for the models if it is the case. This is the subsystem which will identify, finally, a given pronunciation. The complexity of this module depends on the type of identification that is required. For instance, a word recognizer system will be more complex than one that recognizes letters or phonemes.

3.6. Applications of ASR systems

Although in the last years great advances have been achieved in the ASR systems, it is necessary to take into account that with the current technology systems still have a little error rate, so the main applications in which those systems are more successful are those with a simple use and with a certain error tolerance. In addition to these considerations referred to the application itself, the ASR system also have some technologic requirements. For working in real applications, the ASR system needs to have the ability to recognize words or commands in a context of continuous speech, maintain a good behavior despite user changes, presence of noise, etc., and be able to work in real time, among other requirements.

The field of ASR systems applications is big and diverse but basically, with the current technology, there are three fields in which these systems have a bigger impact. These areas are:

- Control systems
- Telecommunication services
- Data input and database access systems

One of the most immediate applications of ASR systems is the help to physical disabled people. Using oral commands it is possible to control many of the daily functions and activities. Some examples of these technologies, some of them still in phase of development, are the wheelchair controlled by voice, hospital beds, telephone oral control and oral activation of domestic devices and systems.

The oral activation of domestic devices and systems, included inside the home automation field, has the aim of controlling them with oral commands. It is possible to control devices like the TV, the HIFI equipments, open and close doors and blinds, turn on / off the heating or the lights, etc. There are several commercial systems in this field which normally offers the possibility of controlling all these devices using the phone line. Recognition systems used in this type of applications use to be isolated word-based with the capacity of rejecting external strange sounds.
One of the areas with more potential applications is telecommunication systems. In certain services added to the phone line, the use of oral interfaces allows an effective reduction in the service costs. Some example of these applications could be the automation of services with the telephone operators and the validation of payments done with a credit card. The incorporation of oral interfaces has also allowed increasing the number of services provided by a telecommunications network. Some examples of these applications are the information services and bank transactions, voice interactive telephone services and information access services. Regarding to the mobile telephony in vehicles, ASR systems have been introduced for allowing controlling the phone (call, answer...) using oral commands.

Another recent application, is the oral writing machine, a voice-text convert system with an extend vocabulary which can transcribe natural speaking to text. This type of systems is being developed currently and it is already possible to find some of them commercialized. Today, it is also possible to find in the markets products like telephones, toys, pocket diaries, etc which incorporate a simple ASR system for controlling the most elemental functions of these products.
4. Gaussian Mixture Models

One of the most important tasks when working with mixture model-based clustering is precisely, selecting the type of function which offers a better adjustment to the data field and the type of task we face up to. Between the different types of mixture model-based clustering, one of the most commonly used is clustering based in Gaussian Mixture Models (GMMs). They are the main option for the work with biometric systems (like speech recognition systems) due to their flexibility and their capability of representing a large class of sample distributions. Also, this type of distributions presents two main advantages versus other type of density functions. Firstly, its ability to form smooth approximations to arbitrarily shaped densities. The second big advantage is that GMMs combine the flexibility of non-parametric methods with the robustness and smoothness of the parametric Gaussian model.

Techniques based on GMM are applied to many different tasks. Some of the most common applications are speaker identification, speech recognition, image segmentation, biometric verification or detection of image color and texture.

In the case of speech recognition systems GMMs are used to represent the speech units (phrases, words, phonemes...). More specifically, the distribution of feature vectors which represent these units is modeled by a Gaussian mixture density.

With a mixture model-based approach to clustering, it is assumed that the data to be clustered are from a mixture of an initially specified number \( K \) of groups in some unknown proportions. In other words, the mixture model is defined as a weighted sum of \( K \) density functions:

\[
p(x|O) = \sum_{i=1}^{K} \alpha_i g(x|\theta_i) \tag{4.1}
\]

Where each \( \alpha_i \) is the mixing probability and each \( \theta_i \) is the set of parameters defining the \( K \) components. As being probabilities, the \( \alpha_i \) must satisfy

\[
\alpha_i > 0, \ i = 1, ..., K \quad \text{and} \quad \sum_{i=1}^{K} \alpha_i = 1 \tag{4.2}
\]

As working with GMM, density function \( g \) will be obviously Gaussian. Then, each mixture component will be expressed like:
Each one with its own parameters, \( \theta_i = \{\mu_i, \Sigma_i\} \). Where each \( \mu_i \) and \( \Sigma_i \) correspond, respectively, to the mean vector and the covariance matrix of the Gaussian \( i \) (for simplicity, from now on, we won’t refer to the covariance matrix but the vector composed by the values of the diagonal which we will call \( \sigma \)). The parameter \( D \) refers to the data dimension.

Covariance matrices can be full rank or diagonal and this property will depend on the type of data that will be used for building the model. In the case of speech recognition systems it is common the using of diagonal matrices due to the low values presented by the cross correlation of the feature vectors in which voice samples are parameterized. There are empirical evidences [18] that diagonal matrices have a better behavior than full rank and that they allow achieving equivalent results in density modeling.

The complete Gaussian mixture model will be defined through its parameters, which are the mean vectors, covariance matrices and mixture weights from all component densities. All these parameters are collectively represented by the notation:

\[
O = (\alpha_1, \ldots, \alpha_K, \theta_1, \ldots, \theta_K)
\]

(4.4)

For characterizing the model it will be necessary to make some choice about the model configuration like the number of components, if covariance matrices are full or diagonal, the dimensions of the density functions, etc. This choice is often determined by the amount of data available for estimating the GMM parameters and how the GMM is used in a particular biometric application and will determine the behavior and the aspect of the model.

The following figures show different examples of GMM. In figure 8 we can see a two-dimensional GMM with two components and parameters \( \mu_1 = [1 \ 2] \), \( \mu_2 = [-3 \ -5] \), \( \sigma_1 = [2 \ 5] \), \( \sigma_2 = [1 \ 1] \) and \( w_1 = w_2 = 0.5 \).

In figure 9 we can see a two-dimensional GMM with 4 components and parameters \( \mu_1 = [1 \ 0] \), \( \mu_2 = [-3 \ -2] \), \( \mu_3 = [0 \ -5] \), \( \mu_4 = [7 \ -4] \), \( \sigma_1 = [2 \ 1.5] \), \( \sigma_2 = [2 \ 1.5] \), \( \sigma_3 = [2 \ 3] \), \( \sigma_4 = [2 \ 2] \), \( w_1 = w_2 = 0.25 \), \( w_3 = 0.2 \), \( w_4 = 0.3 \).
Figure 7. Gaussian Mixture Model with two components

Figure 8. Gaussian Mixture Model with four components
In figures 9 and 10, we can see an example of how a set of features can be modeled using a GMM [23]. Specifically, this example represents a system whose aim is to recognize the sound of one musical instrument between three different options.

Figure 9 shows the representation of the feature vectors representing each musical instrument.

![Figure 9. Feature vectors of three different sounds](image)

Now, given training vectors and a GMM configuration (as mentioned before, number of Gaussians - three -, dimensions - two -, etc.) we wish to estimate the parameters of the GMM which in some sense best matches the distribution of the training feature vectors.

There are several techniques available for estimating the parameters of a GMM. Here, we will focus on one of the most popular, the Expectation Maximization algorithm. The aim of this algorithm will be to find the model parameters which maximize the likelihood of the GMM given the training data. The basic idea of the EM algorithm is, as it will be seen in the next section, beginning with an initial model \( \theta^{t-1} \), to estimate a new model \( \theta^t \), such that \( p(X| \theta) \geq p(X| \theta^{t-1}) \).

Figure 11 shows the Gaussian Mixture Model for the three instruments calculated from the feature vectors shown above. Here, each one of the Gaussians is related with one of the instruments.
Once the mixture model has been calculated, a probabilistic clustering of the data into the three clusters can be obtained in terms of the fitted posterior probabilities of component membership for the data.

An outright assignment of the data into K clusters is achieved by assigning each data point to the component to which it has the highest estimated posterior probability of belonging.
5. Expectation Maximization Algorithm

Mixture Models are typically fitted using EM algorithm. This algorithm makes an iterative estimation of model parameters. For the realization of this task, it starts from an initial guess and then, it proceeds iteratively in two steps: the first one is called the Expectation step (E-step) and the second one is called the Maximization step (M-step).

Given a set of $N$ observations

$$X = \{x_1, x_2, \ldots, x_n\}$$  \hspace{1cm} (5.1)

Where each $x_i$ is a $D$-dimensional vector measurement, we wish to assign a cluster for each observation and to find the parameters of all clusters in such a way that the likelihood (the goodness of fit of the current distribution against the observation dataset) is maximum.

If it is assumed that these vectors are identically distributed, with a distribution $p$, the resulting density for the samples is given by the equation,

$$P(X \mid O) = \prod_{i=1}^{N} p(x_i \mid O) = l(O \mid X)$$  \hspace{1cm} (5.2)

where $l$ is called the likelihood function.

As it is shown in the previous equation, the likelihood function depends directly on the parameters of the model so, the final goal of the algorithm will be maximizing the likelihood through finding the parameter vector $O$ that makes $l$ to be a maximum. In order to estimate $O$ it is typical to introduce the loglikelihood function (5.3), because its maximization is analytically easier.

$$L(O \mid X) = \log \left( l(O \mid X) \right)$$  \hspace{1cm} (5.3)

This transformation will not change the results because since $\ln(x)$ is a strictly increasing function, the value of $O$ which maximizes $l$ also will maximize $L(O \mid X)$. 
5.1. Application to Gaussian Mixture Model

The mixture-density parameter estimation problem is probably one of the most widely used applications of the EM algorithm in the computational pattern recognition field. In this case, we assume the probabilistic model shown in section 4:

\[ p(x_i \mid O) = \sum_{i=1}^{K} \alpha_i g(x \mid \mu_i, \Sigma_i) \]  

(5.4)

where function \( g \) is a Gaussian density, which, as seen before, is given by the expression:

\[ g(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{D/2} \Sigma^{1/2}} e^{-\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu)} \]  

(5.5)

In this case, the log-likelihood expression for the data set \( X \) is given by

\[ L(O \mid X) = \log \prod_{i=1}^{N} p(x_i \mid O) = \sum_{i=1}^{N} \log \left( \sum_{j=1}^{K} \alpha_j g_j(x_i \mid \theta_j) \right) \]  

(5.6)

However, this expression is difficult to optimize, which, as seen before, is the final aim of the algorithm. For simplicity, it is typically used a “trick” that consists of considering the data set \( X \) as incomplete and supposing the existence of a set of unobserved data \( Y=\{y_i\}_{i=1..K} \). The values of this “hidden” property will indicate which density function is responsible for the generation of each particular data item. The union of the data set \( X \) and \( Y \) will be called the “complete-data set”.

Considering this new data set, the aim of the algorithm will be finding the expected value of the likelihood for the complete-data set with respect to the unknown data \( Y \) given the observed data \( X \) and the current parameter estimates.

With this in mind, it is established an auxiliary function called \( Q \), which is defined like

\[ Q(O, O^{(i-1)}) = E[log p(X, Y \mid O) \mid X, O^{(i-1)}] \]  

(5.7)

Where \( O^{(i-1)} \) are the parameters used to evaluate the expectation and \( O \) are the new parameters to be optimized in the current iteration.

In order to maximize the likelihood we compute an estimate \( O \) such that,
\[ L(O) > L(O^{(i-1)}) \]  
(5.8)

Or, which is the same, we maximize the difference,

\[ L(O) - L(O^{(i-1)}) \]  
(5.9)

We build \( Q \) in such a way that it can be expressed as a bound for this difference, then,

\[ L(O) - L(O^{(i-1)}) \geq Q(O, O^{(i-1)}) \]  
(5.10)

The result of this is that instead of maximizing the likelihood directly, we’ll try to maximize function \( Q \), which is analytically easier and will end up in the same results.

With this new point of view, the two steps of the algorithm will consist of:

- \textbf{E-step:} compute the expected value of the complete log-likelihood, conditioned on the data and the current parameter estimate.

- \textbf{M-step:} estimate the new parameters maximizing function \( Q \). Or which is the same, calculate

\[ O^{(i)} = \text{argmax}_O Q(O, O^{(i-1)}) \]

Considering the complete data set, the log-likelihood will follow the next expression

\[
L(O|X,Y) = \log(P(X,Y|O)) \\
= \sum_{i=1}^{N} \log(P(x_i|y_i)p(y)) = \sum_{i=1}^{N} \log(\alpha_{yi}p_{yi}(x_i|\theta_{yi}))
\]  
(5.11)

Applying the previous expression to the definition of function \( Q \) and developing it we can obtain the next expression for function \( Q \) (a complete mathematical development can be consulted in [3]).

\[
Q(O, O^{(i-1)}) = \sum_{k=1}^{K} \sum_{n=1}^{N} \log(\alpha_k) p(k|x_n, O^{(i-1)}) \\
+ \sum_{k=1}^{K} \sum_{n=1}^{N} \log(p_k(x_n|\theta_k)) p(k|x_n, O^{(i-1)})
\]  
(5.12)

As it can be seen above, the function \( Q \) can be expressed as a function of two independent terms.

The first term of the equation depends on the mixture weights (\( \alpha \)) and the second one depends of the parameters of the Gaussian mixture (\( \theta \)). So, for maximizing \( Q \) we can maximize each term separately (again, the complete development is shown in [3]).
The final estimates of the new parameters in terms of the old parameters are as follows:

\[ \alpha_i = \frac{1}{T} \sum_{n=1}^{N} \Pr(i \mid x_n, \theta) \]  \hspace{1cm} (5.13)

\[ \mu_i = \frac{\sum_{n=1}^{N} \Pr(i \mid x_n, \theta)x_n}{\sum_{n=1}^{N} \Pr(i \mid x_n, \theta)} \]  \hspace{1cm} (5.14)

\[ \sigma_i^2 = \frac{\sum_{n=1}^{N} \Pr(i \mid x_n, \theta)x_n^2}{\sum_{n=1}^{N} \Pr(i \mid x_n, \theta)} - \mu_i^2 \]  \hspace{1cm} (5.15)

Where, the sub-index \( i \) refers to the \( i \)th Gaussian density of the mixture and the probability \( \Pr \) is the “a posteriori” probability (the probability that data \( x_n \) belongs to Gaussian \( i \), given the set of parameters \( \theta \)), whose value can be computed applying the Bayes’ rule and is expressed like

\[ \Pr(i \mid x_n, \theta) = \frac{\alpha_i g(x_n | \theta_i)}{\sum_{k=1}^{M} \alpha_k g(x_n | \theta_k)} \]  \hspace{1cm} (5.16)

Both the mean and the covariance matrix are calculated in a similar way as how we would compute a standard empirical average and covariance except that the contribution of each data point is weighted by the value of \( \Pr \).

The mean will be a D-dimensional vector, covariance will be a matrix with DxD terms (as it is a diagonal matrix, values calculated here correspond with the values of the main diagonal).

In the M-step, these equations must be computed following this order as they depend on the previous one. First, the \( K \) new weights will be computed, then the \( K \) new means and finally the \( K \) new covariances.

- A graphical example

In the next three figures an example of classification made with the EM algorithm is shown. In the first place figure 11 shows the graphical representation of a random distribution of bi-dimensional data.
The aim of the experiment will be classify all these input data in several clusters. As the model parameters are initially unknown, we will use the algorithm to build the Gaussian distribution model. For doing this task, the first step will be to decide in how many clusters we want to classify our data.

Firstly, we will execute the algorithm indicating that the desire number of clusters is two. The result thrown by the algorithm will be the one shown in figure 12.
As it can be seen above, the input data have been grouped together in two different clusters. Each of these clusters corresponds with a Gaussian distribution. In addition to classify the samples, the algorithm allows to calculate the parameters of each Gaussian. The values of the model parameters are shown in table 1.

![Figure 12. Classification of a set of random samples in two clusters](image)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Gaussian 1</th>
<th>Gaussian 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>$[-0.4841\ 1.9982]$</td>
<td>$[-2.9702\ -5]$</td>
</tr>
<tr>
<td>Covariance (diagonal)</td>
<td>$[7.6695\ 0.6924]$</td>
<td>$[1.0411\ 1.0304]$</td>
</tr>
<tr>
<td>Weight</td>
<td>0.6667</td>
<td>0.3333</td>
</tr>
</tbody>
</table>

Table 1. Results of applying the EM algorithm to a random set of data

If, on the contrary, we decide that an accurate classification of the data will be done setting the number of clusters (or Gaussians) to 3, the obtained results will be those shown in figure 13.
In this case, as can be seen above, the top Gaussian of the previous example has been divided in two different Gaussians. This time, the algorithm throws the following results for the Gaussian parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Gaussian 1</th>
<th>Gaussian 2</th>
<th>Gaussian 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>([-2.9762, -5.015])</td>
<td>([2.0496, 1.9808])</td>
<td>([-3.0105, 1.9923])</td>
</tr>
<tr>
<td>Covariance (diagonal)</td>
<td>([0.9703, 1.0138])</td>
<td>([1.7696, 0.4863])</td>
<td>([1.0021, 1.0238])</td>
</tr>
<tr>
<td>Weight</td>
<td>0.3333</td>
<td>0.3307</td>
<td>0.3363</td>
</tr>
</tbody>
</table>

Table 2. Results of applying the EM algorithm to a random set of data
5.1.1. Convergence and initialization

The initialization of the model parameters can be a sensitive issue in EM due to the algorithm convergence to a local maximum of the likelihood function, so the final values can be affected by the initial values. For this reason, instead using a random values selection K-means algorithm will be implemented for the initialization of GMM in each cluster.

K-means is one of the simplest unsupervised learning algorithms for solving clustering problems. Its aim is classify n observations in K clusters in which each sample belongs to the cluster with the nearest mean. The algorithm operates performing the following steps:

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

5.1.2. Summarizing the algorithm

In the previous sections the Expectation Maximization algorithm has been explained in detail. However, all this previous development can be condensed into a few steps. Basically, in EM algorithm, the new model becomes the initial model for the next iteration and the process is repeated until some convergence threshold is reached. The algorithm behavior can be summarizing in the following steps:

1. Giving and initial value to the parameters. As mentioned before, this task will be done by executing the K-means algorithm.
2. Evaluating the initial likelihood with the parameter values obtained in the initialization.
3. E-Step: can be reduced to the evaluation of the posteriori probability.
4. M-Step: the new model parameters are estimated by maximizing the expectation of the likelihood.
5. Re-evaluate the likelihood and check for convergence comparing the new $L$ with the previous one.

Steps 3 and 4 are repeated iteratively until the likelihood doesn’t increase anymore.
6. GMM Classifier

The GMM Classifier developed is part of a major speech recognition system. This previous system was implemented in C++ language and it allows the user to choose between several classification algorithms. The new tool adds the option to select a classification based on Gaussian Mixture Models.

6.1. Development environment

For the implementation of the GMM classifier the development environment used has been Visual Studio 2008, so, if some code modification is desired it will be necessary to work with that or posterior versions of the program.

Also, a correct execution will require having installed MySQL Server and Microsoft DirectX SDK (the program will ask for the libraries Microsoft.DirectX.DirectSound.dll and MySql.Data.dll if they are not present).

6.2. Corpora

As it has been explained in previous sections of this document, one of the first steps in an ASR system is the model construction using a set of training data. In this case, a corpus composed by a collection of feature vectors will be used. Each of these feature vectors is related with one of the 37 different phonemes that the system can identify and they will be the data for the construction of the Gaussian Mixture Model, where each mixture will represent one of the phonemes.

This corpus is built from 16425 polish language recordings, both male and female voices. Utterances are designed to cover as much as possible of the space of polish phonetics.

6.3. General description of the tool

6.3.1. Structure

The Speech recognition system is composed by two projects: SpeechRecognition2 and SpeechRecognition2GMM.
The first one is implemented in C++ and is the existent project. The second one is the developed project and has been written in C#.

SpeechRecognition2GMM project is composed by three files (GaussianPdf.cs, EMAlgorithm.cs and GMMClassificator.cs) which purpose and behavior will be explained in later sections of this document.

GMM project is created as ClassLibrary, so when it is built it will produce a 'dll' file, SpeechRecognition2GMM.dll.

6.3.2. Inputs

The program inputs will be a set of audio files, recorded in WAV format.

As it was explained in section 3, the speech signals contained in these audio files will be processed so their feature vectors will be obtained. Then, these feature vectors will be compared with the model previously build with the aim of classify them into one of the existent classes.

6.3.3. GMM Classifier

The idea in which the classifier is based is that the similarity of feature vectors from one audio file and the model is expressed by the product of the Gaussian mixture density, i.e. the likelihood.

The implemented classifier will work following the next steps:

The EM algorithm is executed taking as input data the values content in the training corpus. These values can be considered as vectors which represent each of the 37 phonemes of the polish language. It is considered that each one of these phonemes can be represented by a Gaussian density function with a concrete mean and covariance. Therefore, the purpose of the execution of EM will be obtain the parameters of each Gaussian and in this way having characterized the clusters in which we want to classify the data.

When an input file is selected, the values representing of each phoneme will be obtained. These values will be organized in segments. What will be done in the program, is calculating the probability of each segment belongs to each Gaussian (phoneme). This probably is precisely the “a posteriori” probability whose mathematical expression is shown in equation (5.16). Once these probabilities are calculated, they will be ordered in decreasing order in such a way that we finally obtain a list in which the values in the first positions represent the most probable options for that audio.
For the realization of all of this, 3 different classes have been implemented. They are described as follows.

- **GaussianPdf**

  This class is used for creating an object representing a vector Gaussian density. This value will be obtained through the implementation of the mathematical expression seen in equation 4.2. However, due to the use of diagonal covariance matrices this expression can be simplified and expressed like

  \[
  g(x|\mu, \Sigma) = \prod_{i=1}^{D} g(x|\mu_i, \Sigma_i) = \frac{1}{2\pi^{D/2}\prod_{i=1}^{D} \sigma_i} e^{-\sum_{i=1}^{D} \frac{(x_i-\mu_i)^2}{2\sigma_i}}
  \]

- **EMAlgorithm**

  In this class the execution of the EM algorithm is carried out with the aim of obtaining the parameters of the different mixtures that compose the GMM. The data used for this task will be the corpus mentioned before. However, it is important to note that before processing the data we need to calculate their Discrete Cosine Transform (DCT) and remove the first component (DC component). Then, from an original set of vectors with length 11 we have obtained another with length 10 which we will work with from now on.

  The implementation of the algorithm will be done following the steps showed in section 5, i.e. giving an initial value to the starting parameters (the mean will be initialized through the execution of K-means algorithm, variance will be initialized directly from its mathematical expression using the previously computed mean values and the weight is assumed equal for every Gaussian), computation of the initial value for the likelihood, iterative execution of E-Step and M-Step until a convergence threshold is reached. The E-Step and M-Step will be executed inside a loop which needs a stop condition. Theoretically, the iterations should stop when the likelihood calculated in the current iteration is equal to the one calculated in the previous iteration. However, in practice such precision is not necessary so the condition established for skipping the loop will be that the difference between both likelihoods is less than a certain tolerance. Once the algorithm execution is ended, it will have been created a list in which each phoneme is represented by a Gaussian with certain weight, mean and covariance, i.e. the GMM will have been created.
• **GMMClassifier**

It is the main class since it is the point of the program where the classification is indeed made. The main method in this class receives as input argument the segment to classify. Again, the DCT of the input values will be obtained, and with this information and the values obtained through EM algorithm, it will calculate the probability of that vector belongs to each Gaussian. Then, it will create a phoneme list ordered from the most probable to the least.

### 6.3.4. General working of the program

When running the program, it launches a window like the one in figure 14.

![Inicial window](image)

*Figure 14. Inicial window*

The first time the program is executed, it will be necessary to indicate the folder that contains the audio files which we would want to use as the application input. Once this folder have been selected, the audio files will be shown in a list like the one shown in figure 15.
The next step will be the program configuration. For doing that, we will select “Konfiguracja” (Configuration) menu and we will go to “Ustawienia” (Settings) tab. There, it will be necessary to select where the data base is located.

The last step in the configuration settings will be to select the classifier we will use. As it was commented before, the program will offer us different options for the classification (Itakura-Saito, K-nearest neighbor, Gaussian Mixture Models...). All these options will be shown in “Klasyfikacja” (Classification) tab. In our case, we will select, between the several classifiers offered, the option “GMM Classifier”.

Figure 15. Initial window with the list of possible input files
Now that the program is configured, we will simply select the audio file we want to analyze and click in “Analiza” (Analysis) button.
7. Experiments and results

In the previous section the designed tool have been explain. Now it is required to test its behavior in different situations. Throughout this section different types of tests will be proposed and its results will be shown.

In the first place the Phoneme Recognition Rate (PRR) will be tested. This rate measures the proportion of correctly classified feature vectors in a testing set of data. For calculating, we will select randomly a vector set for the training phase and another set for the testing phase. It is important to note that since it is a test for checking the goodness of the classification we already know the phoneme assigned to each vector in both data sets so we can compare the results with the reality.

Once we have executed the EM algorithm using the first vector set (training set) we will have build the GMM, then, we will have a set of means, covariances and weights, where each triple represents the parameter set of one Gaussian mixture. Afterwards, we will assign each of the vectors of the second set (testing set) to one cluster (Gaussian). For doing this task, we will calculate the ‘a posteriori’ probability that one vector belongs to each cluster and we will select for that vector the cluster with the biggest probability. Once we have all vectors classified we will compare the assignation with the cluster we know each vector belongs.

We are doing this test several times varying some parameters. These parameters will be the number of mixtures and the number of coefficients in the DCT. We will do the tests for a number of N={1,2,3,4,5} mixtures and for M={3,5,7,10} DCT coefficients. As a result we will obtain the following data set.

The next two tables show, respectively, the mean and de typical deviation of the previous set of data.
8. Conclusions
9. Bibliography


[20] *Speech Recognition.*
http://www.learnartificialneuralnetworks.com/speechrecognition.html


[23] Yi-Chieh W. *Gaussian Mixture Model*. http://cnx.org/content/m13205/latest/