TECHNICAL UNIVERSITY OF CRETE

ELECTRONIC AND COMPUTER ENGINEERING DEPARTMENT

DIPLOMA DISSERTATION

MUSIC GENRE CLASSIFICATION USING TEMPORAL AND SPECTRAL FEATURES

by

MARKAKI ANTHI

CHANIA
AUGUST 2008

Acknowledgments

Abstract

In recent years, electronic music archives are gaining popularity. The variety and amount of songs, even in personal computers, are being increased, which creates the need for effective methods for music classification and retrieval. The task of organizing and selecting music becomes more and more challenging. So, in the music domain, there is the need to characterize the genre of a track from the information that exists in its content. Musical genres are categorical descriptions that are used to characterize music in music stores, radio stations and on the internet.

A typical way of music classification is the recognition of songs, based on header tags. However, it is quite difficult to keep tag organization in the whole archive. For that reason, to determine the music genres of a music piece, three basic stages of a pattern recognition system have been used: frame feature extraction, training of a classifier and classification. A number of content-based features are first extracted. The signal is split into frames and certain characteristics of the music signal within the frames are represented with a feature vector. After the features are extracted from each music track, a classifier is necessary to determine the genre of the music track. Then, the trained classifier is used to assign the feature vectors of the test data.

In this thesis, we demonstrate the effectiveness of music genre classification based on extracted features, applying various configurations. Finally, a simple program is implemented to produce a playlist from a classification chart. The application employs Self-Organizing Maps. It provides an overview of a music collection by navigating on the map and showing the artist and the name of the song, based on the location of the cursor. It gives to the user the opportunity to draw a circle on the music landscape. The music map consists of grid squares, where each of them contains a small number of songs. When the user selects a region in the map, the system finds all the songs included in this region and put them in a playlist. Alternatively, the user can click on a point on the music landscape and select a

number of songs that he wants to listen. When the user selects a point in the map, the system finds the n songs that correspond to the closest points and put them in a playlist. This way, the user is able to see how similar the pieces are. It is known that the recognition of musical genres is not a trivial task even for humans. Therefore, the playlist gives us an estimation of how good the classification is.

The results are quite promising. The MFCC's can be used in music genre classification with an accuracy of 65purposes. Comparing our work with similar projects, we infer that MFCC's provide the best accuracy, which could be improved in combination with other features.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Scope

In recent years, there has been a great proliferation of multimedia databases, which create the need for affective methods for classification and retrieval of data. With the development of computer networks, it becomes more and more popular to download digital music from the internet. Personal computers and portable digital music players are platforms for storing and playing music. As the music collections grow, the task of organizing and selecting music becomes more and more challenging. Therefore, effective management of a large digital music database is a substantial matter. So, in the music domain, there is the need to characterize the genre of a track from the information that exists in its content. Musical genres are categorical descriptions that are used to characterize music in music stores, radio stations, and on the internet. Although the division of music into genres is considered to be arbitrary, there are some criteria that can be used for this purpose. Some of these are the texture, the instrumentation and the rhythmic structure of the pieces. People are good at genre classification. We are able to judge genre without constructing high level descriptions. However, nowadays with the advance of technology, the categorization does not be performed manually. Several techniques for automatic genre classification has been discovered, which lead to the development of audio information retrieval systems for music.

To determine the music genres of a music piece, three basic stages of a pattern recognition system have been used: frame based feature extraction, training of a classifier and classification. A number of content-based features are first extracted. First, the signal is split into frames, and certain characteristics of the music signal within the frames are represented with a feature vector. In general, the features that are used in music classification can be categorized into three classes: the timbral texture, the rhythmic features and the pitch content features. After the features are extracted from each music track, a classifier is necessary to determine the genre of the music track. Finally, the trained classifier is used to assign the feature vectors of the test data. For each class, the recognition rate is calculated as a percentage of correctly classified pieces among all the classified pieces, and the overall recognition rate is calculated as the arithmetic mean of recognition rates of the individual classes.

The Self-Organizing Map (SOM), an unsupervised neural network, is used to cluster the pieces of music. It provides a mapping from high-dimensional music spaces to two-dimensional maps where similar pieces of music are located close to each other.

In order to see how well the music pieces are located in the map, a play from the map application has been created. This application employs Self-Organizing Maps. It provides an overview of a music collection by navigating on the map and showing the artist and the name of the song, based on the location of the cursor. It gives to the user the opportunity to draw a circle on the music landscape. The music map consists of grid squares, where each of them contains a small number of songs. When the user selects a region in the map, the system finds all the songs included in this region and put them in a playlist. Alternatively, the user can click on a point on the music landscape and select a number of songs that he wants to listen. When the user selects a point in the map, the system finds the n songs that correspond to the closest points and put them in a playlist. This way, the user is able to see how similar the pieces are. It is known that the recognition of musical genres is not a trivial task even for humans. Therefore, the playlist gives us an estimation of how good the classification is.

1.2 Outline

The rest of the dissertation is organized as follows. In the next chapter, short information is provided about the work that has been done in music genre classification. In Chapters 3 and 4, there are descripted the features that are extracted as well as the procedures of clustering and classification. In Sections 5 and 6, our work is presented, the implementation results and an attempt of comparison with other published results. Finally, we conclude in chapter seven and discuss some issues about future extensions.

Chapter 2

Related work

2.1 A Comparative Study on Content-Based Music Genre Classifiacation

A lot of work has been done in feature extraction for music genre classification from Tao Li, Mitsunori Ogihara and Qi Li [7].

Various content-based features have been used. One of this kind is the timbral textural such as the mel-frequency cepstral coefficients, the spectral centroid, the spectral rolloff, the spectral flux, the zero crossings and the low energy which are useful for the differentiation of similar sounds. Other kind is the rhythmic content feature like the beat that is extracted form the beat histogram and catches the rhythmic pulse from music signals. In order to construct the beat histogram an onset detection curve is computed, where the successive bursts of energy correspond to successive pulses. Then, on this curve is performed an autocorrelation function and lastly a peak picking to find the tempo. Pitch content features which describe the melody and harmony of the signals, are calculated via pitch histogram. It contains the peaks of the autocorrelation function which is computed to the envelopes of each frequency band. Since it is desirable to have local and global information about the sounds, wavelet transform is being used. The wavelet transform provides information about a signal that is not apparent in the original form of the signal. The original signal is in the time domain. The wavelet transform reveals frequency information and information about the times at which different frequencies occur.

A common way to implement the wavelet transform is the discrete octave band decomposition [8]. The signal is decomposed by applying a low pass and a high pass filter. At each step of the decomposition the signal obtained from the the low pass, is recursively decomposed. More specifically, the wavelet transform can be thought as a window that is shifted along the original signal. At each location, the wavelet is correlated with the signal at that particular point and the process is repeated. So, the wavelet is stretched to a larger scale. A large scale corresponds to a low frequency while a short scale corresponds to a high frequency. The process continues with larger and larger wavelet scales. The wavelet result of the process is a map of correlation values, called wavelet coefficients. This way, it has computed the wavelet histogram.

The wavelet filters that are used are the Daubechies, which are the most common used in music and image information retrieval systems. The histogram is constructed at each subband. The first three moments of each subband are kept and finally is computed the subband energy.

For the classification has been used support vector machines, k-nearest neighbor classifier, Gaussian mixture models and linear discriminant analysis. The support vector machines are learning machines that perform classification. They construct a N-dimensional hyperplane that separates the data into two categories, the positive data and the negative data. They are based on the concept of decision planes that define decision boundaries. A decision plane separates a set of objects that have different class memberships.

The K-nearest neighbor is a clustering method used to separate the data with similarities between various classes. It is based on the euclidian distance between a test sample and the training samples. For each row of the set that is to be classified, the k closest members of the training dataset are located. The euclidean distance measure is used to calculate how close is the target row that is examined from each member of the training set. This procedure is repeated for all the rows.

Gaussian mixture models consist of local Gaussian modes and an integrated network. They try to describe a system using combination of all the Gaussian clusters. It is assumed to exist for each class a probability density function. The expectation maximization algorithm is used to estimate the parameters of each Gaussian model. The expectation step assumes the cluster parameters to be correct and finds the most likely distribution of the data. The maximization step assumes the distribution to be correct and maximizes the likelihood of the cluster parameters.

Linear discriminant analysis is a classic method for classification. It finds a linear transformation that discriminates among classes and performs classification. The main idea is to describe the main decision boundary instead of describing the properties of each class. This means that we dont want to describe the class but the separating surface.

Two datasets have been used. The first contains ten genres with 1000 songs each genre. The ten genres are Blue, Classical, Country, Disco, Hip Hop, Jazz, Metal, Pop, Reggae and Rock. The second dataset contains 756 songs of five genres. The genres are Ambient, Classical, Fusion, Jazz and Rock. The results are shown in tables 2.1 and 2.2.

Features	**SVM**	**GMM**	**LDA**	**KNN**
DWCHs	74.9	63.5	71.3	62.1
MFCC	58.4	46.4	55.5	53.7
Beat	26.5	22.1	24.9	22.8
Pitch	36.6	25.8	30.7	33.3

Table 2.1: The accuracy of the learning method for the dataset 1

Features	**SVM**	**GMM**	**LDA**	**KNN**
DWCHs	71.48	64.77	65.74	61.84
MFCC	60.45	53.43	59.26	59.93
Beat	43.37	37.95	40.87	41.27
Pitch	37.56	29.62	37.82	38.89

Table 2.2: The accuracy of the learning method for the dataset 2

2.2 Combining audio and symbolic descriptors for music classification from audio

Similar work has been done by Thomas Lidy, Andreas Rauber (University of Technology, Austria) and Antonio Pertuse, Jone Manuel Inesta (University of Alicante,Spain). Rhythm patterns have been used such as a short time fourier transform and a discrete fourier transform applied to a sonogram. A rhythm histogram is constructed containing the modulation amplitude values of the critical bands that have been computed from the rhythm pattern. Then, the mean, the median, the variance, the skewness, the kurtosis, the minimum and maximum values are calculated for each band, and constitute a Statistical Spectrum Descriptor (SSD). An onset detection algorithm is applied to extract information about the rhythm of the signal. The onset detection calculates the minimum, the maximum, the mean, the median and the standard deviation of the distance of frames between successive onsets. Moreover, the number of onsets is added to the onset features. For the classification it has chosen to use the linear Support Vector Machines, with the Weka machine learning software. Three different datasets were used for the genre classification and the accuracy was 66.71

2.3 MARSYAS

Marsyas (Music Analysis Retrieval and Synthesis for Audio Signals) is a free software framework for audio analysis, synthesis and retrieval. It is considered to be a useful tool for audio analysis, providing a lot of algorithms.

Marsyas is a framework for audio signal processing, music information retrieval and classification written in C++. It provides many tools. The software performs both training and classification. The procedure is composed of pre-filtering, feature extraction and classification.

The pre-filtering section includes different filtering techniques. After that, the feature extraction is performed to the signals. A lot of different features have been supported to this system, some of which are fast fourier transform analysis, short time fourier transform, spectral and temporal centroids, Linear Prediction Analysis, Mel-frequency cepstral coefficients and many others. In most cases, combination of these features or derivatives of them are used. For the feature extraction the first thirty seconds and the last fifteen seconds are omitted in order to avoid undesirable noises. TheMel-frequency cepstarl coefficients are derived from the fourier transform of the audio signals adopted to the mel scale, which is a scale that corresponds to the human hearing. After applying a discrete cosine transform, the coefficients are produced. The short- time spectral features are calculated from the fast fourier transform of each short analysis window and are based on the magnitude spectrum S(f). Most specifically, we have:

2.3.1 Timbre Analysis

- **Centroid:** It is the center of gravity or second moment of the spectrum. It is calculated from the following equation where $S(f)$ is the magnitude spectrum.
$$X = \frac{\sum_{f=1}^{M} S(f) \times f}{\sum_{f=1}^{M} S(f)} \tag{2.1}$$

- **Rolloff:** It is the frequency below which 85obtained. It shows the density of low frequencies in the signal. It is calculayted from the following calculation
$$\sum_{f=1}^{x} S(f) = 0.85 \times \sum_{f=1}^{M} S(f) \tag{2.2}$$

- **Flux:** It shows the difference between spectra of subsequent time frames. It is calculated from the following equation
$$X(f) = \sum_{f=1}^{M} (S(f) - S(f-1))^2 \tag{2.3}$$

- **Zero crossings:** It is a count of the number times the signal crosses from positive to negative.

2.3.2 Rhythm Analysis

The rhythm is a useful information for music classification. The goal is to define the tempo. The tempo declares the speed that repetitive sounds occur. In order to find the repeated patterns that define the rhythm, the beat histogram is constructed. After decomposing the signal using the discrete wavelet transform and recomposing with the inverse wavelet transform, different frequency bands are produced. Each of these bands, are undergoing envelope extraction and all these envelopes are summed. Then, an autocorrelation function is performed to the signal to see how well it matches with itself. Finally, a peak picking finds the best peaks and puts them to a histogram.

2.3.3 Pitch Analysis

The pitch is computed from a pitch histogram. As with the beat histogram, the highest peaks are selected and are computed the distance between the two peaks and the amplitude of the highest peak.

2.4 Automatic genre classification of music content: a survey

N.Scaringella and G.Zoia reached an automatic genre classification of music approach [11].

The first kind of feature that was extracted is the timbre which makes two sounds with the same pitch and loudness sound different. In this category belong temporal features that are computed from the audio signal frame, energy features that refer to the energy content of the signals. In this case, root mean square energy of the signal are calculated, as well as the energy of the harmonic component of the power spectrum and the energy of the noisy part of the power spectrum. Another kind is the spectral shape features which describe the shape of the power spectrum and the centroid, the spread, the skewness, the kurtosis, the slope, the roll-off frequency, the variation and the mel- frequency cepstral coefficients are calculated. Furthermore, melody and harmony analysis has been done. Melody declares the succession of pitched events wile harmony declares the pitch simultaneity in music. The melody description more specifically can be found in [5]. Two functions that characterize the pitch distribution have been used. The first contains information about the pitch range of the piece and the second that maps all pitches to a single octave. Three different techniques extract the rhythm, like the autocorrelation function of features over time, the fast fourier transform in order to estimate the modulations of features and the histogram of onset intervals. All these features are extracted from 30-seconds pieces of songs, after the beginnings of the pieces. This aims at the avoidance of the introductions that in most cases do not represent the category that the pieces belong to.

For the classification, have been used both supervised and unsupervised methods [2],[10]. The simplest way for the unsupervised learning method is the similarity measures. The simplest measure is the Euclidean distance or a cosine distance. However, these measures have sense only if the features are time-invariant. To make a time-invariant representation, statistical models are appropriate. Typical models are the Gaussian mixture models [4][5] and the Hidden Markov Models (HMMs) that model the relationship between features. K-means algorithm is another way of clustering but its drawback is that the centroids have to be known in advance. Finally, the Self-Organizing Map (SOM) and the Growing Hierarchical Self- Organizing Map (GHSOM) are used to cluster the data. These are unsupervised artificial neural networks that map high dimensional input data onto lower dimensional output spaces. Nevertheless, the topological relationships between the input data are being kept. A big number of supervised classifiers are used to classify unlabelled data. The k-nearest neighbor algorithm is based on the idea that the neighbors influence the decision. The k closest vectors of the training set are selected and the feature that is to be classified is assigned the label of the most represented class in the neighbors. For the Gaussian mixture models, it is assumed the existence of a probability density function. The expectation-maximization algorithm is used to estimate the parameters for each Gaussian component and the mixture weights. Hidden Markov Models (HMMs) are used to model the different duration possibilities of music. The Linear Discriminant Analysis (LDA) finds a linear transformation that discriminates among classes and makes classification with some metrics such as Euclidean distance. Support Vector Machines (SVMs) manage a margin maximization and a nonlinear transformation of the future space.

For the experiments a dataset which is composed of 1515 songs over 10 genres was used. The ten genres are Classical, Ambient, Electronic, New-Age, Rock, Punk, Jazz, Blues, Folk and Ethnic. From all the songs, the 1005 were used for the training while the 510 for the testing.The results are shown in table 2.3.

	Ambient	**Blues**	**Classical**	**Electronic**	**Ethnic**	**Folk**	**Jazz**	**New-Age**	**Punk**	**Rock**
Ambient	52.94	0	0	7.32	4.82	0	0	26.47	0	5.95
Blues	0	76.47	0	0	0	4.17	0	0	0	3.57
Classical	2.94	0	100	0	8.43	0	0	0	0	0
Electronic	5.88	0	0	53.66	6.02	4.17	4.55	5.88	0	19.05
Ethnic	2.94	0	0	7.32	59.04	12.5	4.55	20.59	0	0
Folk	0	5.88	0	1.22	3.61	62.5	0	2.94	0	2.38
Jazz	0	3.94	0	3.66	6.02	4.17	81.82	8.82	0	5.95
New-Age	29.41	0	0	4.88	4.82	8.33	4.55	32.35	0	5.95
Punk	0	0	0	0	0	4.17	0	0	100	4.76
Rock	5.88	14.71	0	21.95	7.23	0	4.55	2.94	0	52.38

Table 2.3: The results of the dataset

Chapter 3

Feature Extraction

Feature extraction is an essential pre-processing step to pattern recognition and machine learning problem. It is often decomposed into feature construction and feature selection. Feature extraction is the core of content-based description of audio files. With feature extraction, we extract from the signals information that represent the music. This way, a computer is able to recognize the content of a piece of music. The features must be comprehensive, must not require a big amount of storage and not much computation as well. Many different features can be implemented for music classification, but we are focused on content-based acoustic features. Many of them are highly interdependent, this means that are based on the same initial computations. Timbral textural features, the category that the mel-frequency cepstral coefficients belong to, are useful for us to distinct sounds that have similar rhythm and pitch contents. Rhythmic content features such as the beat and the tempo, are used to show the movement and the regularity of the rhythm of music pieces. Pitch content features are used to describe the melody of a song.

The analysis of the signal, gives us a global description of the sound. But we are interested in the dynamic evolution of the feature. So, the analysis has to be done with the use of a short-time window. Usually, we apply a Hamming window which removes edge effects. This way, signals are divided into frames (the position of each window), that are statistically stationary.

One of the main challenges when designing music information retrieval systems is to find the most descriptive features of the system. There is a great deal of features. Timbral features are used to distinct the sounds with the same or similar rhythm [6]. Some of these are the zero crossings which is an appropriate measure of noisiness, the centroid which consists a measure of spectral sharpness and brightness, the roloff which measures the spectral shape, the flux which computes the difference between the magnitudes of successive distributions, and the MFCCs which consist a compact representation of the spectrum of an audio signal. It takes into account the human perception of pitch.

Rhythmic features describe the movement of the signal over time. Instead of just measuring the tempo, is more interesting for classification purposes to extract information about rhythmical structure and beat strength. A beat histogram is a curve describing beat strength as a function of a range of tempo values. Peaks of the histogram correspond to the main beat and other sub beats.

Pitch features describe the melody and the harmony of music signals. It is extracted via pitch detection techniques, the most common of whom is the pitch histogram. It contains information about the amplitudes and periods of maximum peaks of it.

Timbral features are calculated for every short-time frame of sound, while rhythmic and pitch features are computed over the whole file. This means that timbral features focus on the statistics of local information but from a global perspective and rhythmic and pitch features since are computed over the whole file, dont contain enough information for classification purposes.

3.1 MFCCs

The use of Mel-Frequency Cepstral Coefficients (MFCCs) for music information retrieval has become standard since a lot of years. MFCCs are dominant features which are used in speech recognition systems, such as the systems which recognize numbers spoken into a telephone. They are used for recognizing people from their voices and for genre classification as well.

They are short time spectral features. The mel-frequency cepstral coefficients algorithm is based on transformation from time domain to frequency domain and filtration with perceptual filterbank. Firstly, for each frame, is calculated the fast fourier transform and the result is stored in a vector F. The result of this operation is being filtered with each filter from Mel filterbank and the result is aggregated and stored in a vector S. After that, the logarithm of the vector S is being calculated.

Finally, this vector is being transformed by discrete cosine transform [9]. The whole procedure is described by the following equations.

- $$F(i) = [\Re(F(i))]^2 + [\Im(F(i))]^2 \tag{3.1}$$

- $$S(k) = \sum_{i=0}^{N/2} (F(i) \times M(i)) \tag{3.2}$$

- $$L(m) = \log S(k) \tag{3.3}$$

- $$C(n) = \sum_{i=0}^{L-1} L(i) \times \cos(\frac{\pi n}{2L} \times (2i+1)) \tag{3.4}$$

Where F is the result of the FFT, k is every filter, C is the Mel frequency cepstral coefficients and n the coefficient number of M coefficients. The m MFCC features are organized in a fxm matrix, where each row consists of the m MFCC values for a frame and there are f rows, the number of frames into which the signal has been segmented.

The reason that is used the mel scale is because many experiments have shown that the ears perception to the frequency components in the speech does not follow the linear scale but the mel- frequency scale, which should be understood as a linear frequency spacing below 1KHz and logarithmic spacing above 1KHz. The common used formula that reflects the relation between the mel frequency and the physical frequency is given by the following equation where f is the frequency in hertz.

$$M(f) = 1125 \times \log(1 + \frac{f}{700}) \tag{3.5}$$

The relation between Mel and Hertz scale is shown in figure 3.1.

Thus, the MFCC's are calculated as follows [4]:

- Divide signal into frames
- For each frame, obtain the amplitude spectrum
- Convert to Mel spectrum
- Take the logarithm

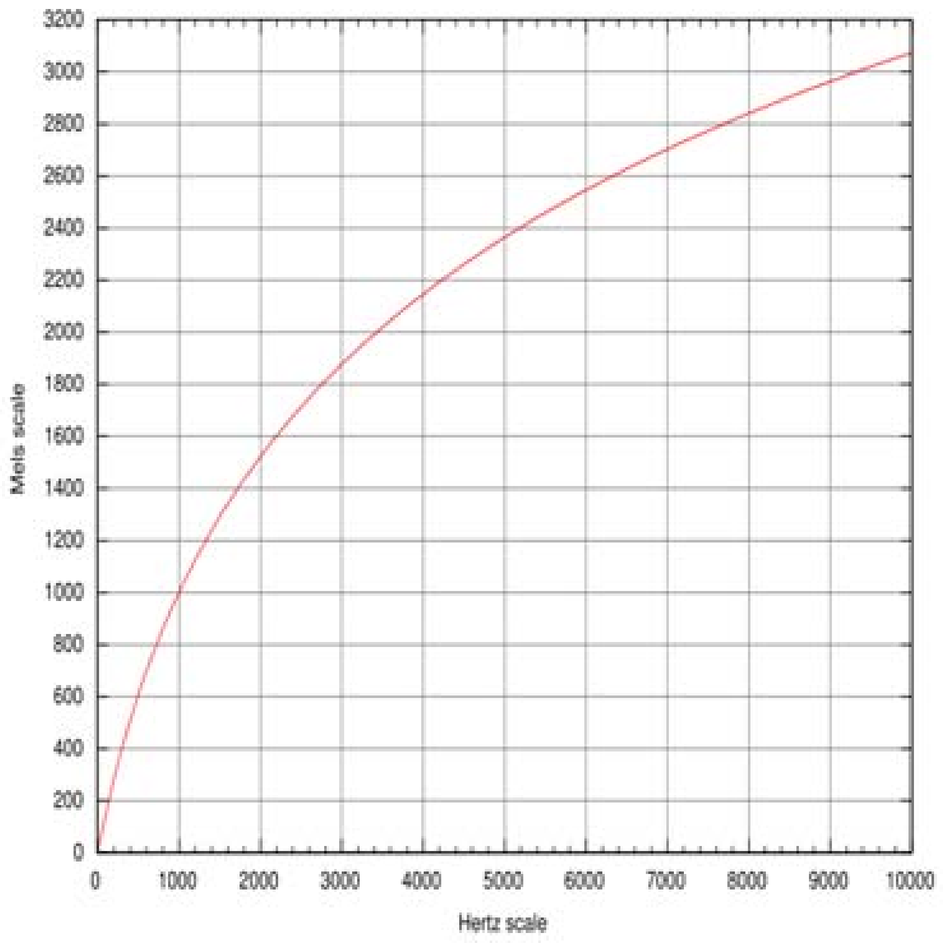

Figure 3.1: The relation between Mel and Hertz scale

- Take the discrete cosine transform (DCT)

3.2 PITCH

Pitch is a feature that has been underrated for many decades. This feature is important for recognizing a womans voice from a mans. However, it is quite difficult to understand its perception.

Pitch is an auditory attribute of tones, and any tone is physically characterized by a frequency. For that reason, it is immediately dependent on frequency. However, even and isolated tones have multiple pitches. So, there is not a direct relationship between pitch and frequency. Moreover, in real life, most sounds are accompanied by additional sounds. As a consequence, a model of pitch perception must include the segregation of tones from noise and other multiple tones. Lastly, no one tone can be characterized by just one frequency. For all these reasons, it is clear that the pitch detection techniques are not accurate.

The pitch content features describe the melody and harmony information about music signals and are derived from the pitch histogram. In order to construct this histogram, the audio signal is first decomposed into a number of octave frequency bands using discrete wavelet transform.After that, we calculate the autocorrelation funvtion of the log of spectrum of the signals. Spectrum autocorrelation method has been successfully used in several pitch estimations. The idea is derived from the observation that a periodic signal has a periodic magnitude spectrum, the period of which is the fundamental frequency. The autocorrelation function has a local maximum in the position corresponding to the period of the spectrum.

So, we extract the fundamental frequency since we know that the first minimum of the autocorrelation function corresponds to the fundamental period. The inverse of the fundamental period gives us the frequency that we want. This way, we have the pitch value. All these pitch values are entering into a histogram. The largest entry in the histogram is taken to be the pitch.

The goal is to estimate independent pitch estimates as separate frequency bands and then combine the results to yield a global estimate. This solves several problems, one of which is inharmonicity. The higher armonics may deviate from their expected spectral positions, and even the intervals between them are not constant at narrow enough bands. Thus we utilize spectral intervals to calculate pitch likelihoods at separate frequency bands, and then combine the results in a manner that takes the inharmonicity into account. Another advantage of bandwise processing is that it provides robustness in the case of badly corrupted signals, where only a fragment of the whole frequency range is good enough to be used.

Spectral and cepstral analysis, need to divide the signal into a series of frames. Conventional methods usually employ a fixed-length hamming window. However, windowing a speech using a fixed- length window may yield following two drawbacks. First, a frame contains both periodic and non- periodic parts of speech signal and secondly the periodic signal may be cut off at an unbefitting point, which

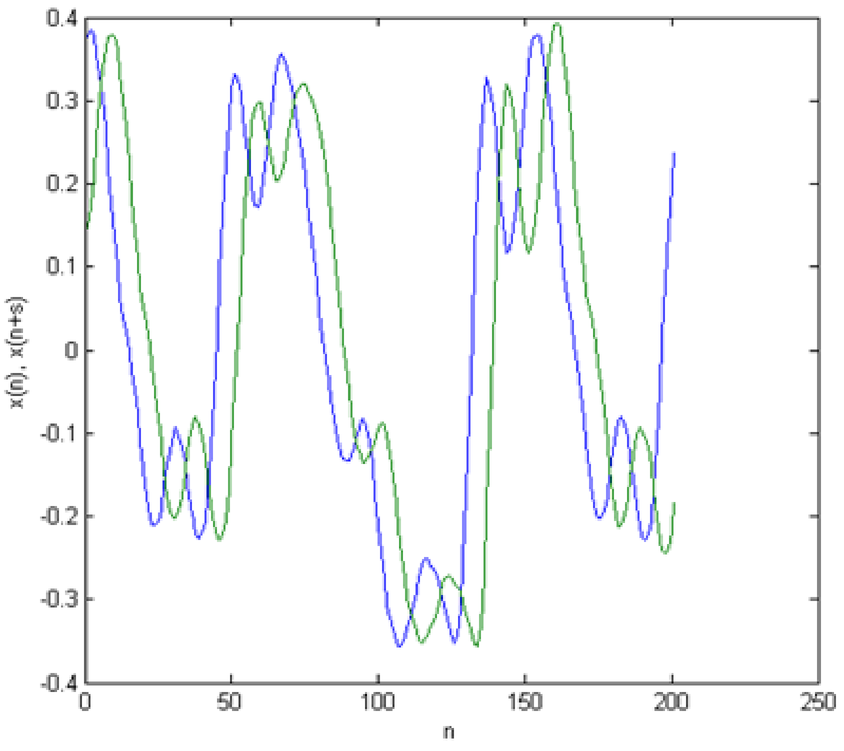

Figure 3.2: The original signal and the shifted signal

makes the period of signal incomplete. Due to that non-periodic part have negative effect on harmonic information of speech signal, which leads to spectral distortion in frequency domain. Feature extraction from fixed-length windowed frame cannot reflect the nature of signal. Pitch synchronous analysis which intercepts signal using a window whose length is integral times of the pitch period of the speech, provides a solution to the problem of harmonic leakage.

3.3 BEAT

The beat features try to count the number of beats of a song. They are driven by instruments that operate in the lower frequencies, like the drum or the bass.

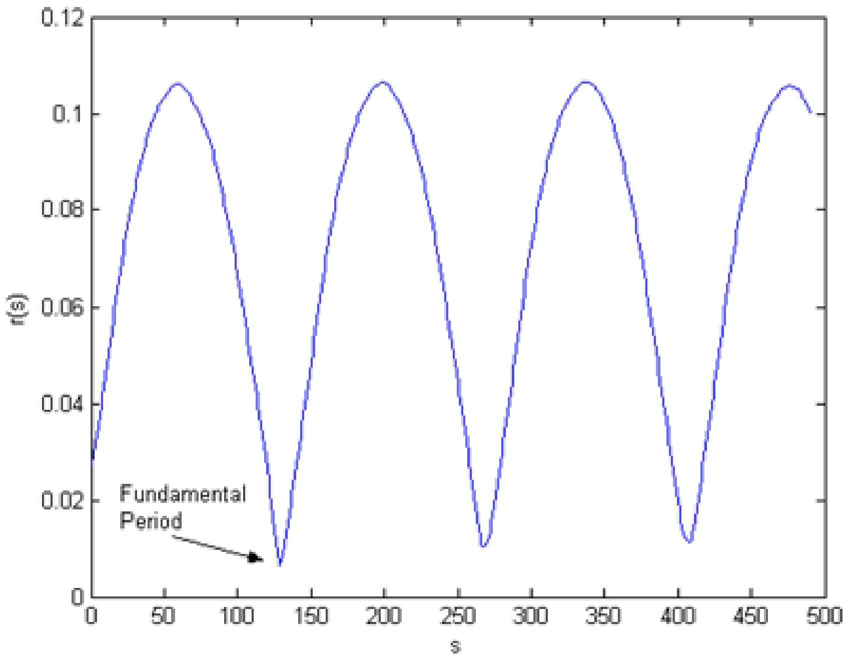

Figure 3.3: The autocorrelation function

Beat estimation is the process of predicting the music beat from a representation of music, symbolic or acoustic. The beat is assumed to represent what humans perceive as a binary regular pulse underlying the music. The beat in music is often marked by transient sounds. The rhythmic content features are extracted from the beat histogram [14]. The beat histogram describes how much periodicity is in the audio excerpt at different tempo levels, in many cases, the most prominent peaks correspond to the main tempo of the excerpt.

In order to construct the beat histogram, the envelope of each band is extracted separately [12]. For each band, full-wave rectification, low pass filtering, down-sampling and mean removal are performed in order to extract an envelope. The low pass filtering will cut off teh frequencies higher tahn 200Hz. The envelpopes of each band are summed up and the autocorrelation function is calculated to capture the periodicities in the signal's envelope. Some onsets positions may correspond to the position of a beat, while other onsets fall of the beat. By detecting the onsets in the acoustic signal, and using this as input to a beat induction model, it is possible to estimate the beat[15]. Autocorrelation involves comparing a signal with versions of itself delayed by successive intervals. This yields the relative strength of different periodicities within the signal. In terms of musical data, autocorrelation allows one to find the relative strength of different rhythmic pulses. Such histograms are sometimes be used directly as features. The dominant peaks in the autocorrelation function are accumulated over the whole signal into a beta histogram. The tempo corresponds to the main beat.

Some of the demands of a beat estimation system are stability and robustness. Stability to ensure that the estimation is yielding low errors for music exhibiting stationary beats and robustness to ensure that the estimation continues to give good results for music breaks without stationary beats. In addition, the system should be casual, and instantaneous. Casual to ensure real-time behavior, and instantaneous to ensure fast response.

The beat estimation has been evaluated by comparing the beat per minute (BPM) output of the algorithm to a human estimate. The human estimate was found by tapping along while the musical piece was playing, and finding the mean time difference between taps. The estimated BPM values match the human estimate in a percent of 60

In figure 3.4 we see the beat histogram that correspond to a classical piece and in figure 3.5 the beat hisogram of a rock music piece. The dominant values (90 for classical and 120 for rock) are the beats per minute, which consist the main tempo.

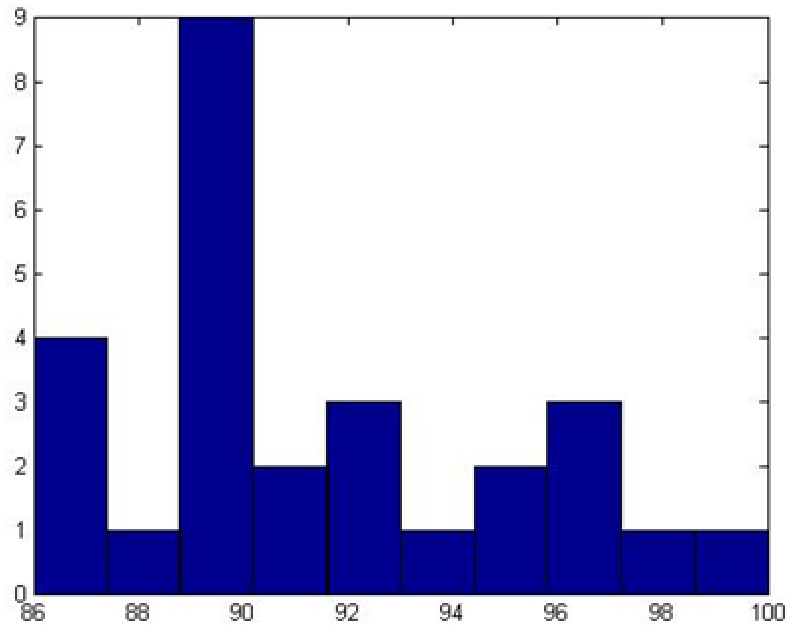

Figure 3.4: The result of a classical piece

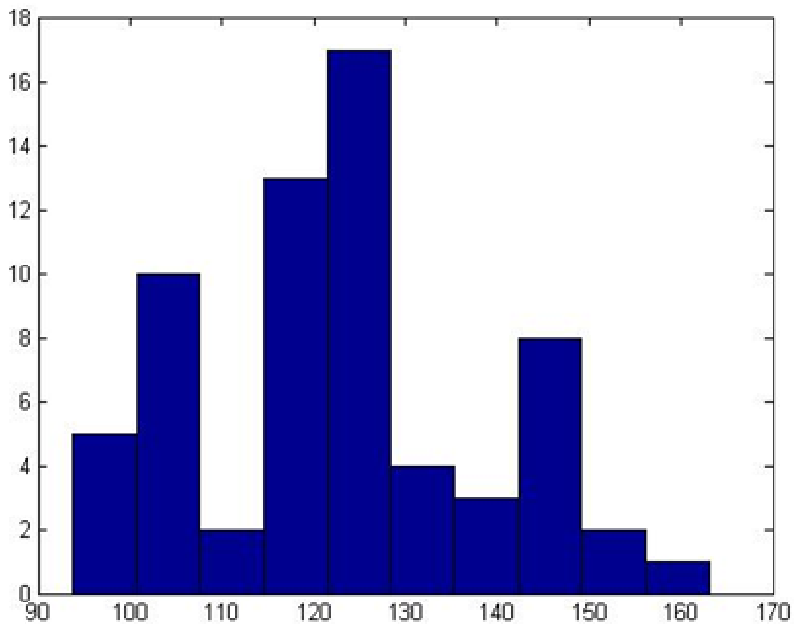

Figure 3.5: The result of a rock piece

Chapter 4

Clustering and Classification

4.1 Clustering

Clustering is an approach to identify natural grouping of similar entries in such sets of unclassified data-often without any a priori knowledge of the similarities that may involve. The principal idea is the partition of the dataset into sub-classes, called clusters. So, cluster is a collection of data objects that are similar to one another and thus can be treated collectively as one group. Clustering has wide applications in pattern recognition, spatial data analysis, image processing, economic science etc. The difficult with clustering is what constitutes it. It can be shown that there is no absolute best criterion for the choice. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit his needs. Nevertheless, there is a number of problems associated with clustering. Among them is the fact that dealing with large number of dimensions and large number of data items can be problematic because of time complexity. In addition to the effectiveness of the method depends on the definition of the distance. Moreover, if an obvious distance measure does not exist, it must be defined which is not always easy, especially in multi-dimensional spaces.

4.1.1 The k-means algorithm

K-means algorithm is an algorithm used to classify objects based on attributes and features into k number of groups. K is a positive integer number. K-means is one of the simplest unsupervised learning algorithms that solve the clustering problem. The main idea is to define k centroids, one for each cluster [1]. These centorids should be placed in an appropriate way because different location could cause different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. The first grouping has completed. Now, the k new centorids that have produced must be recalculated. This loop is repeated until the k centorids do not change their location. This means that they do not move any more. The algorithm is composed of the following steps:

- Specify k, the number of clusters to be generated
- Choose k points at random as cluster centers
- Assign each instance to its closest cluster center using Euclidean distance
- Calculate the centroid (mean) for each cluster and use it as a new cluster center
- Reassign all instances to the closest cluster center
- Iterate until the cluster centers do not change any more

Each instance x in the training set can be represented as a vector of n values, one for each attribute.

$$X = (x1, x2, ..., x3) \tag{4.1}$$

The Euclidean distance of two vectors x and y is defined as:

$$|x - y| = \sqrt{\left(\sum_{i=1}^{n} (x - y)^2\right)} \tag{4.2}$$

The mean μ of a set of vectors C is defined as:

$$\mu = \frac{1}{|C|} \times \sum_{x \in C} x \tag{4.3}$$

However there are some difficulties with this algorithm. Firstly, although the algorithm will produce the desired number of clusters, maybe the centorids will not be representative of the data. Secondly, the method is computationally inefficient. Each step of the procedure requires calculation of the distance between every possible pair of data points and comparison of all the distances. This requires a lot of time. The main drawback of the k-means algorithm is that the cluster result is sensitive to the selection of the initial cluster centroids. The initial choice is of great importance for the whole procedure. If it could be ensured good initial clustering centorids using other techniques, then the k-means algorithm would work properly to find the optimal clustering centers. It is clear that in order to use this algorithm, the number k of clusters need to be specified. However, for continuous distributions, there exists a set of k principal points for all positive integers k. There is no right or wrong values for k. Instead, the appropriate choice depends on the particular application and must be determined by the investigator.

In figure 4.1 we see two clusters and how the k-means algorithm has located their centroids.

4.2 Classification

Classification is a technique used to predict group membership for data instances. In other words, it is a process of partitioning a set of data or objects in a set of sub-classes. Classification techniques analyze and categorize the data into known classes. Each data sample is labeled with a known class label.

The main difference between clustering and classification is that clustering is an unsupervised learning method while classification is a supervised. This means that classification analysis requires that the user / analyst know ahead of time how classes are defined. The objective of a classifier is not to explore the data to discover interesting segments, but rather to decide how new records should be classified. In classification it is known the class labels and the number of classes, while in clustering we dont know the class labels and may neither the number of classes.

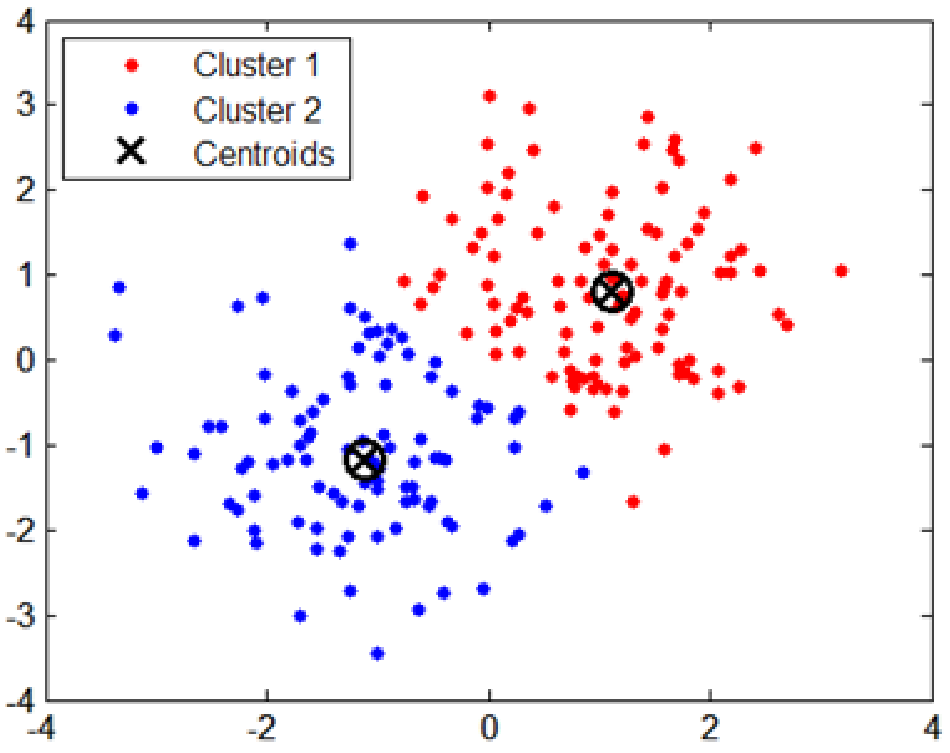

Figure 4.1: The clustering of two centorids

4.2.1 Self-Organizing Map

Artificial neural networks are a promising part of the science community. The Self-Organizing Map (SOM) is one of the most popular artificial neural algorithm for use in unsupervised learning and data visualization [13]. The SOM has been used for a wide range of purposes. First of all, the algorithm was used for the diagnosis of speech voicing. Due to SOMs we can categorize voice samples much more precisely than a man could have done. Other examples of SOM applications in the area of Speech Recognition are speaker identification and isolated-word recognition. This is very useful to recognize isolated and carefully articulated words. Moreover, in engineering applications, the SOM is often used as a look-up-table. Similar to locating the most similar entry in a table for a given input pattern, we determine the best matching unit on the map. Furthermore, SOMs have contributed to the emerging research area of bioinformatics. Using the SOM, you are able to find similarities in expression profiles and suggest inferences about gene functions. SOMs are used to explore financial statements of enterprises and to reveal relationships between the type of corporation and the risk of bankruptcy. SOMs appear to cope very well with the large amount of accounting and financial data available on firms from balance sheets and annual files. Finally, SOMs are useful for natural language processing. They are a very interesting approach to exploratory collections, because relationships between text items, such as similarity, clusters, gaps and outliers can be communicated naturally using spatial relationships, shading and colors.

SOMs are constructed as follows. First, you have to decide the geometry of nodes. The nodes are mapped into k-dimensional space, initially at random, and then are adjusted. Each iteration involves randomly selecting a data point and moving randomly selecting a data point and moving the nodes in the direction of this point. The closest node is moved the most, whereas other nodes are moved by smaller amounts demanding on their distance from the closest node in the initial geometry. Neighboring points in the initial geometry tend to be mapped to nearby points in k-dimensional space [3].

The SOM algorithm is not a clustering algorithm. It is considered as a tool in reducing the dimensionality of the data and for information visualization. It is not a tool that produces an explicit partitioning of a dataset into a precise number of groups. The maps do not show sharp cluster borders and there is no obvious centroid. We can theoretically think of each node on the map as a cluster centroid.

There are different techniques used for the initialization of the map. The first approach is to use random values, independent of the training data set. It is a poor way of initializing the map because it requires quite a number of additional training cycles until the map can be said to be at least roughly representative of the training data. The second approach is to use random samples from the input training data. When the training commences, the map is already in a state in which it represents at least a subset of the input data items. This reduces the number of training iterations needed and lowers the computational cost. Nevertheless, the choice of input samples used for the initialization is random and the number of map nodes is very small compared to the number of training data items. So, the initial map is not likely to be truly representative of the given dataset. The third approach is to reflect the distribution of the data more faithfully. However, this is not easy since the map is usually only two-dimensional while the dataset is often of a much higher dimensionality. If there is one dimension which always holds values very close to its mean, then it would be relatively easy to predict the values of the dimension.

Visualizing a SOM is challenging because the input data is usually of a high dimensionality. By projecting the input space to a two-dimensionality grid we can express the similarity of two samples as the distance between them. One of the simplest ways of coping with higher dimensions is to use a two-dimensional coordinate plane and to incorporate further axes for each additional dimension. Color as a means of representing dimensionality is used in numerous SOM visualizations. Since every color has a red, green and blue component, it is capable of displaying three dimensions at once. At the same time, it appears to be very easy for the human eye to spot similarities in color as opposed to similarities between two geometric figures. Similarity coloring has turned out to be so intuitively comprehensible that it has become the generally accepted method of choice when the objective is to visualize the cluster structure of the data.

To detect the cluster boarders on a map a so-called distance matrix is often helpful. It is shown in figure 4.3. A popular example is the U-matrix (Ultsch and Siemon 1990), which is shown in figure 4.2. It visualizes the distances of each map unit to its immediate neighbors using grey shape. Inside clusters the distance between neighboring units will be small, visualized by a light shade of grey. On cluster boundaries, the differences between neighboring units is large, resulting in a dark shade of grey. Furthermore, there SOMs offer the choice to provide the training set with labels for the different groups.

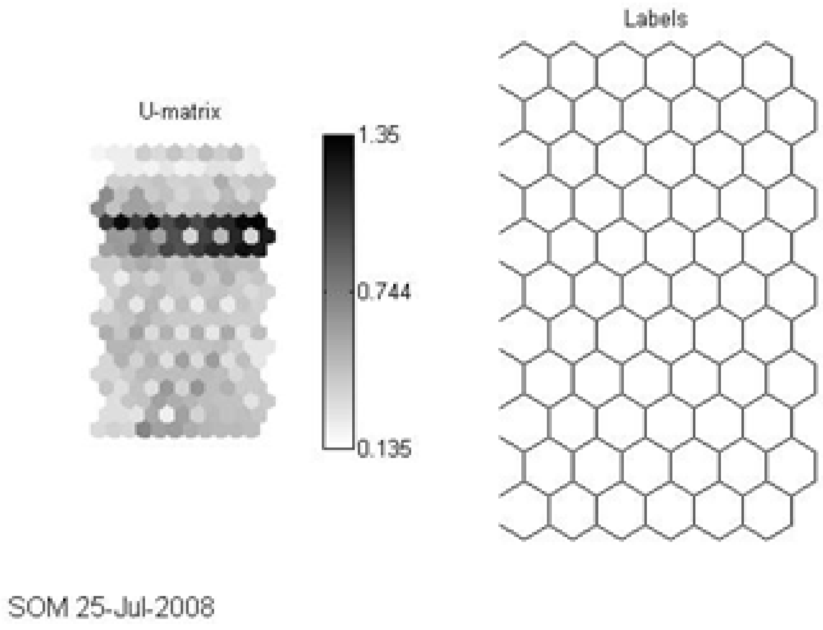

Figure 4.2: The U-matrix

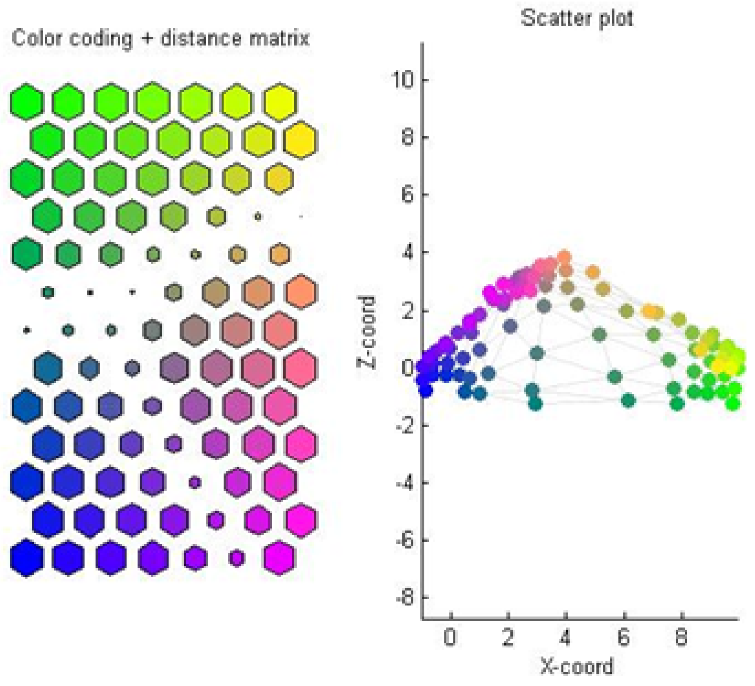

Figure 4.3: The distance-matrix

For the experiments, we used a dataset, which contains 4000 songs over twelve genres. The twelve genres are Classical, Classical-Opera, Dance-Electro, Hard Rock-Heavy Metal, Jazz, Blues, Latin, Punk Alternative, Rock-Pop-Classic, Rock-Pop-Alternative-Other, Vocalists and World-Various. The excerpts of the dataset were taken from mp3 compressed audio files.

First of all, we made k-means clustering per category. The half songs are being used for the training and the rest half for the testing. For the training, the MFCCs of all the songs in each category are being concatenated and in the resulting table is implemented the k-means algorithm. So, a 16x12 table of the centroids is produced. Then, the Euclidean distances for each test song are being calculated and follows the assignment of a category to each song, based on the minimum Euclidean distance. We make the same procedure by using k-means clustering per group. We calculate again Euclidean distances for each test song. Based again on the minimum Euclidean distance, we assign each song to a group and then in the category that the group belongs to. After using self- organizing map we make multidimensional scaling from high-dimensional to low-dimensional space, usually two or three and we organize the music data into clusters. In the first case, we map the groups in each category using groups k-means data and in the second we map the categories using category k-means data. For the visualization of the results, are being used distinct colors.

Chapter 5

Our Approach

5.1 Data preparation

The aim was to collect a database that would cover different musical genres. So, we have twelve genres: classical, classical-opera, dance-electro, hard-rock-heavy-metal, jazz-blues, latin, punk- alternative, rap, rock-pop-classic, rock-pop-alternative-other, world-various and vocalists. The amount of pieces for each genre is not the same and is shown in table 5.1.

Genre	**Number of songs**	**Number of groups**
Classical	148	10
Classical-opera	76	2
Dance-electro	350	15
Hard-rock-heavy-metal	78	5
Jazz-blues	228	14
Latin	258	13
Punk-alternative	358	16
Rap	94	5
Rock-pop-classic	884	36
Rock-pop-alternative-other	1254	66
Vocalists	32	6
World-various	101	3

Table 5.1: The number of songs of our dataset

5.2 Production of wave files

All pieces in the twelve collections were given as mp3-files. To make them readable by Matlab, they were converted to .wav files since the build-in routines that Matlab contains can process .wav files. This was achieved using the lame decoder, an open source tool. Since these mp3 files have been created by extracting from audio CDs, the sampling frequency is 44100 Hertz (Hz), which means that the amplitude value of the audio signal is scanned and stored 44100 times per second. However, for our demands, the processing of wave files and the feature extraction can be done with a lower quality of sound. So, in order to reduce the size of files that are going to be processed, we reduce the sampling frequency at 16000 and 11025Hz. The sampled amplitude values are usually coded with 16 bits per sample, leading to 65536 possible values to describe the amplitude. So, using the lame, with one command and appropriate parameters the mp3 files are undergoing downsampling and new mp3 files are produced. Then, format normalization is applied and each music file is decoded to raw Pulse Code Modulation (PCM) and converted to the .wav format, but with a new sampling frequency 11025 and 16000Hz. Moreover, taking into account the duplication of the data when using stereo sound, it is obvious that the processing of such files is very space consuming. For that reason, it was decided to use only one channel instead of two. As it was seen, there was not any effect on the results. This can be done either with lame or with the Matlab. In order to avoid a huge consumption of hard disk space and a long duration for importing and processing the data into Matlab, we chose to do it with the lame.

In order to make this procedure automatic, we made batch scripts. By executing the batch script from the root folder where the music genre folders reside, every subdirectory is scanned and each mp3 file is processed by the lame commands as described above.

Now the wave files are ready to be processed using custom code written in Matlab. However, before that it is necessary to separate the songs into training set and testing set. Since the Matlab has some restrictions concerning the main memory, it was decided to use only ten seconds of each song. Besides, it is obvious that sometimes the beginning and the end of a piece is not representative of the category that it belongs to. So, we used the seconds after the first sixty seconds. This way, we managed to have less memory occupied during processing and also to succeed better classification results.

5.3 Train

The basic idea is to scan all the subdirectories and to process every wave file that exists. As mentioned above, this program is executed from the root folder of each genre. Using systems commands in the Matlab code, we get into every directory that is found even and the most nested. In each directory, a list of the wave files, that exist there, is created and every wave file is processed with the frontend.The result in the table contains the vectors of the features extracted from the wave files. For each file, this table is stored in a mat file. Next step is to produce the appropriate k-means tables, using similar procedure with systems command in Matlab code. Specifically, in the case that we want to do clustering per category, for each category, all the subdirectories are scanned by opening each mat file that happens to be meet, and the data of the tables are concatenated. At this point, it is important to mention that we dont concatenate all the MFCC vectors of the mat files, but only the rows that correspond to the ten seconds described above. This has to do with the configuration of each scenarios concerning the duration of the frame as defined in the frontend parameters. Moreover, if a song is smaller than two minutes, the ten seconds of the middle of the piece are extracted.

If we want to do training per category, we create the concatenated table that contains the features of all the songs of this category. Respectively, in the case of training per group, we create the concatenated table that contains the features of each group.

The process is completed by applying the k-means algorithm in all these tables, producing 16 clusters (k=16). The resulting k- means tables are stored in mat files. We follow exactly the same procedure when dealing with the beat and the pitch features. The only difference is that in that case, we extract one only value, which correspond to the main peak of the histogram in each case.

5.4 Test

After the training per category or per group, some .mat files have been created that contain the matrix resulting from the k-means algorithm. It is a table of the centroids. For each song that we want to classify, the Euclidean distances between the vectors of the under testing song and the vectors of the k-means table are calculated according to the equations 5.1 and 5.2 (where D is the distance, μ is the mean value, i is the cluster and j the category). We assign each song to the category, from which the sum of the Euclidean distance is minimum. In case of the clustering per group, we assign each song to the group with the minimum distance, and then to the category that this group belong to. The program is executed from the root folder and offers with choice either to test all the songs of all the categories at once, or to test all the songs of a single category. With the creation of appropriate loops, we denote the number of songs that have been classified for each genre and we calculate how many songs were classified correctly, that gives us the desired results. This way, it is possible to create a table with all the results for all the genres and to save them as .mat files for later processing.

$$d_t(x_j) = \sum_{i=1}^{16} d_\epsilon(\vec{\mu_{ij}}\vec{x_t}) \tag{5.1}$$

$$\hat{j} = argmax_j \frac{1}{N} \sum_{t=1}^{N} d_t(x, j) \tag{5.2}$$

5.5 Scenarios

In the following figures (5.1, 5.2, 5.3, 5.4, 5.5, 5.6) we present the results of different scenarios for train and test, using different configurations and different features. Finally, in table 5.2 we present the overall results of the 6 scenarios.

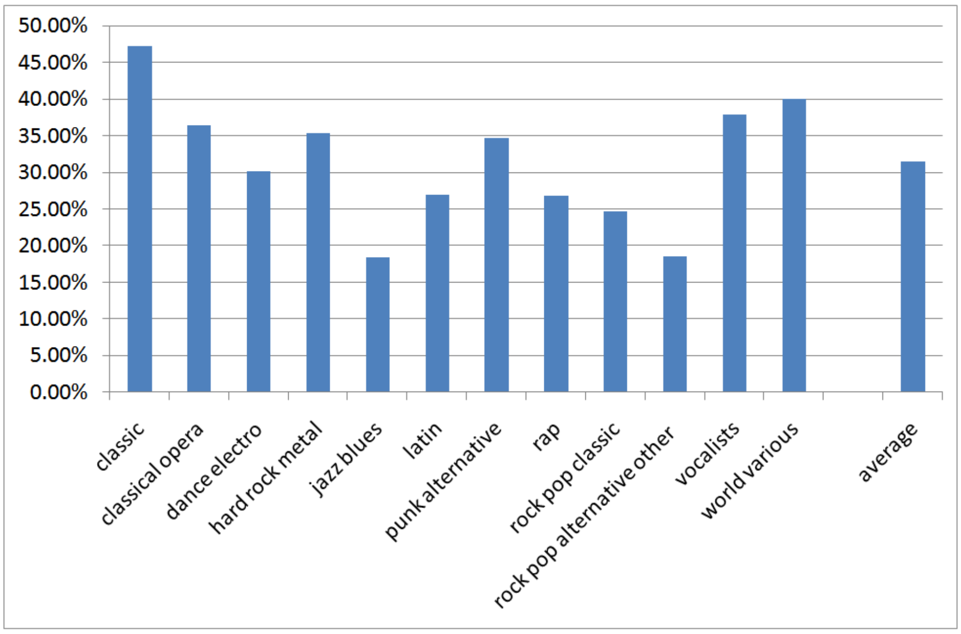

Figure 5.1: Results for beat, k-means per genre, sfr=16KHz, frame size=10ms

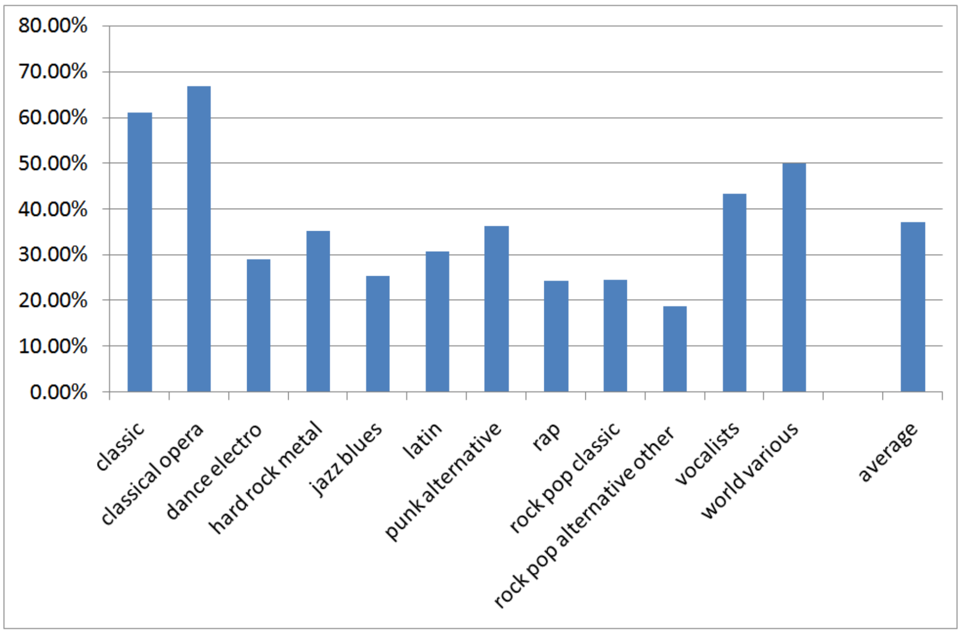

Figure 5.2: Results for pitch, k-means per genre, sfr=16KHz, frame size=10ms

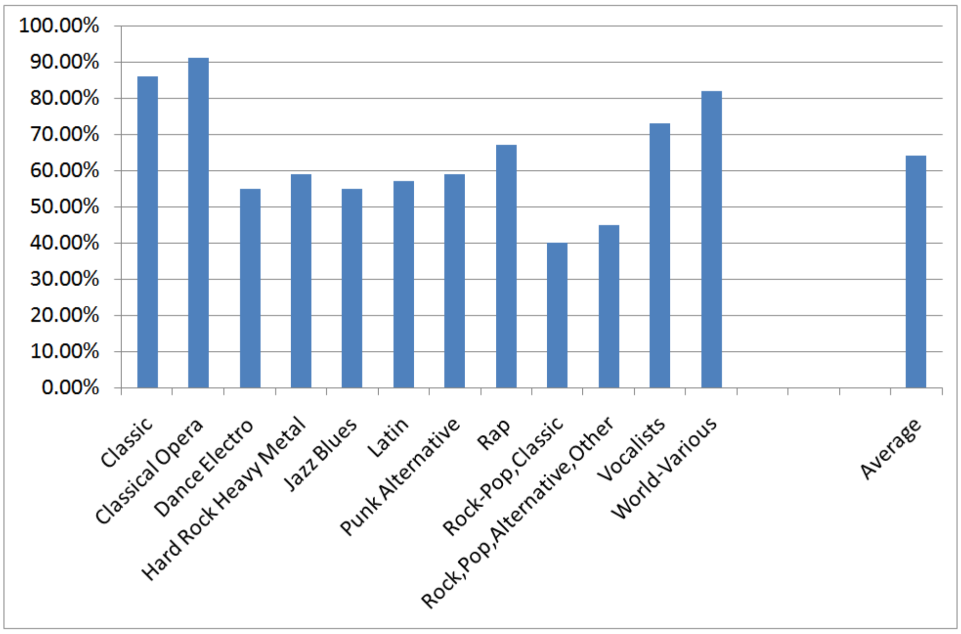

Figure 5.3: Results for MFCC's, k-means per group, sfr=16KHz, frame size=10ms

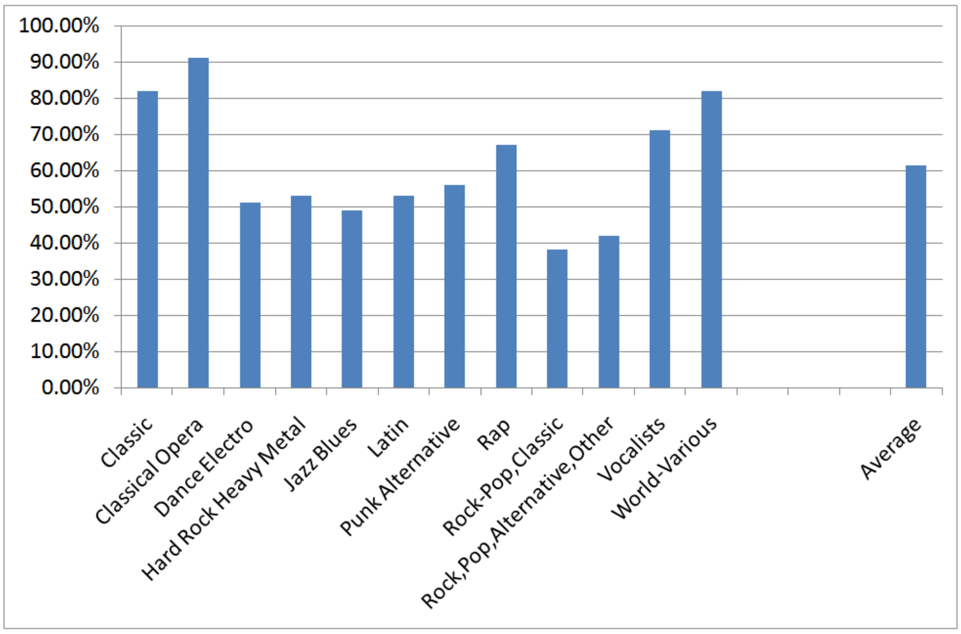

Figure 5.4: Results for MFCC's, k-means per genre, sfr=16KHz, frame size=10ms

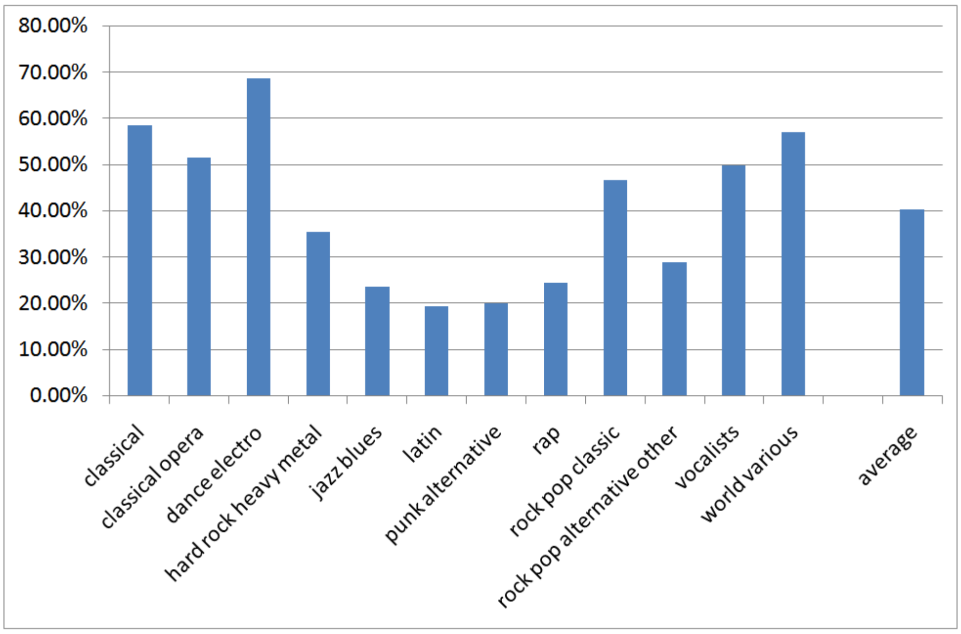

Figure 5.5: Results for MFCC's, k-means per genre, sfr=11KHz, frame size=10ms

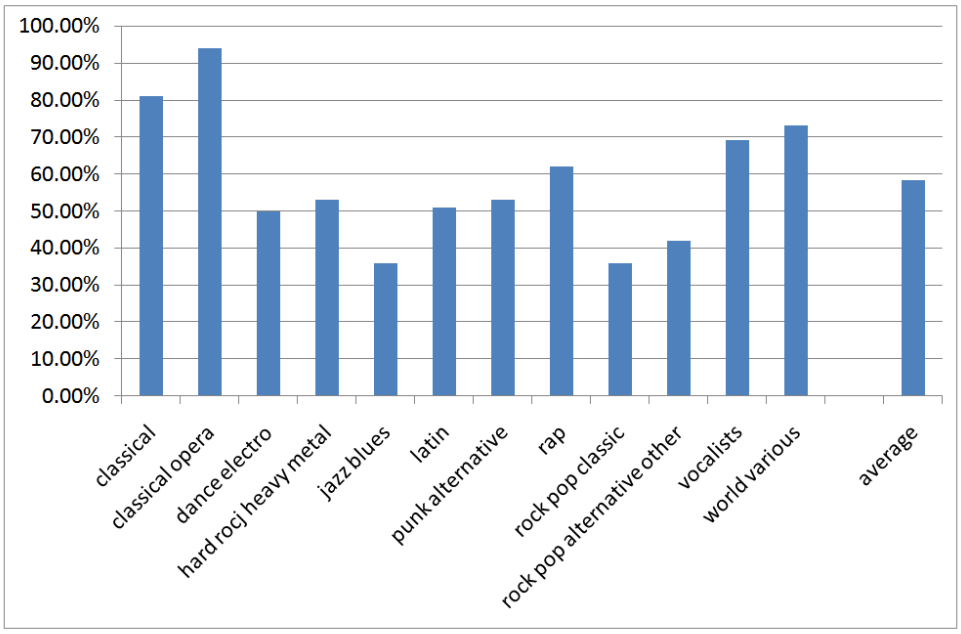

Figure 5.6: Results for MFCC's, k-means per genre, sfr=16KHz, frame size=20ms

Scenario	**Accuracy**
BEAT, k-means per genre, sfr=16KHz, frame size=10ms	31.42%
PITCH, k-means per genre, sfr=16KHz, frame size=10ms	37.10%
MFCC's, k-means per group, sfr=16KHz, frame size=10ms	64.08%
MFCC's, k-means per genre, sfr=16KHz, frame size=10ms	61.25%
MFCC's, k-means per genre, sfr=11KHz, frame size=10ms	40.20%
MFCC's, k-means per genre, sfr=16KHz, frame size=20ms	58.33%

Table 5.2: The overall results

5.6 SOM

Applying the som algorithm we produce the visualization of the data. Even in data with more dimensions, the som algorithm applies multidimensional scaling, resulting to two or three dimensions. To produce this visualization, we implemented two different scenarios. In the first scenarios, we apply the som algorithm on the MFCC values of each song. In the second scenarios, we apply the algorithm on the k-means data of the concatenated MFCCs values of all the songs per genre.

Similar to the train procedure for the feature extraction, this program is also executed from the root folder. After parsing the directory tree in depth, at each MFCC mat file, we extract the features vectors that correspond to ten seconds and concatenate the MFCCs of all the files into a table. At the same time, we create an array of labels, of the same length with the array of the concatenated MFCC values, which contains the name of the genre that each song belongs to. This means that between the two arrays there is a correspondence of rows. In other words, the first row of the labels array corresponds to the first vector of the MFCC array.

By using the function som_data_struct, we create a structure from the array with all the MFCC vectors. At this structure, we add the array with the labels. After normalizing the data of structure, we execute the algorithm of som with the function som_make. The result is a new structure that has undergone the necessary multidimensional scaling and our data have been represented by two coordinates. This way, they can be represented to the plane.

Using the appropriate parameters for the visualization of the results, we use the U-matrix. At this matrix, each component is a distance measure between two adjacent neurons. High values in the U- matrix represent a frontier region between clusters and low values represent a high degree of similarities among neurons on

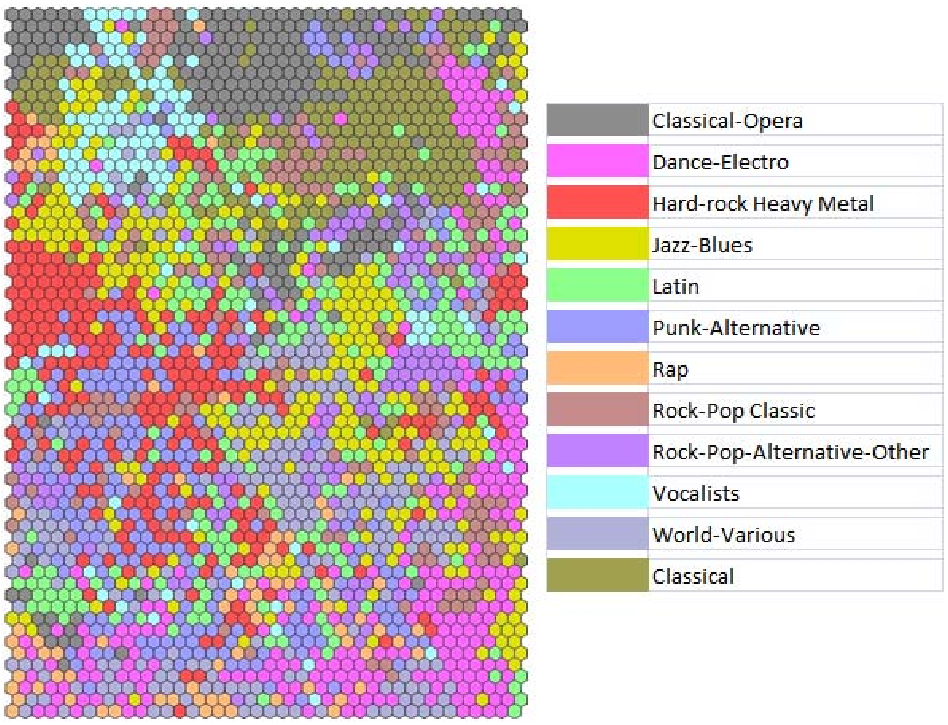

Figure 5.7: The results of our dataset

that region. For each song, the matrix of the values consists of 2000 values. However the structure that results after the som contains a matrix with fewer values because for each cell we have kept the prevalent value.

Moreover, we represent in a cellular map the classification of the pieces that has been produced from the som. The number of cells is the same with the number of resulting data from the som. For that reason, we created a chromatic code, by making a color for each genre, and an array of the same length with the array of labels that we described above. This way, at each row of this chromatic array, we put the color of the category that the point-row of the other array belongs to. So, we have three references for each point: its som data, its label and its color. With the aid of the function som_cplane we create the cellular map.

The results of the SOM algorithm is shown in the figure 5.7:

5.7 map mp3 playlist

As we have mentioned above, in order to evaluate the effectiveness of MFCC features and the visualization that SOM performs, we implemented two applications that give us the possibility to judge acoustically the result. The philosophy of the programs is based on the use of the cellular map that has come up from the process with the SOM algorithm. We used one hundred and twenty songs from all the genres, and with the path name of each song, we made the SOM structure. After the execution of the algorithm, we used the resulting map as an input image and the path name with the coordinates of each song as data input. The coordinates have been computed after appropriate scaling according to the dimensions of image in pixels.

Initially, in both versions, when the program is loaded, it is demanded from the user to give as input the image that is going to be used as map and the corresponding data file. In the first version, the user selects the number of songs that he wants to put in the playlist. As the cursor is being moved on the cellular map, the user can see the song corresponding to that position. Clicking on a point on the map, a program of repeating songs is opened. So, a playlist has been created, which consist of a number of songs that the user has selected. The songs are the n closest to that point that we have clicked on. By clicking on a new point, the playlist is replaced with a new one. In the second version, the user can draw a circle and the playlist wiil consist of the ongs included in the circle.

Both programs have been written in java and the Eclipse has been used as a development application. In the first version, We used a jframe object, into which we implemented a jpanel object. In the jpanel, we implemented a jlabel object, where the image is appeared. At this jlabel, we have implemented three classes to open the image, to open the data file and to select the playlist size. Moreover, we have implemented two methods, one for catching the mouse moves and one for the mouse clicks. So, the main parts of the program are:

- **openDataItemActionPerformed:** It is a class that is used to open the file data. This file, at each row, contains three elements, separated with tabs. The first two are the coordinates x and y, and the third is the mp3 file with his pathname. For the read of this file, it is used the class readfile, which dissociates all the elements of each row, so that we can process them separately.

- **OpenImageItemActionPerformed:** It is a class that is used to open the image file.

- **setPlayListLengthActionPerformed:** it is a class that is used to give the number of songs that we want to put to the playlist. The default value is 1.

- **imageAreaMouseMoved:** It is a j labelmethod that takes the coordinates of the cursor and finds the closest song to that point, by computing Euclidean distances. Moreover, it prints the name and the coordinates of the song

- **imageAreaMouseClicked:** : It is a jlabel method that is used for the creation of the playlist and the opening of the program. When we click on a point, it is created a matrix of all the songs and the distance of them from that point. This matrix is being sorted and the n first songs are selected to be put to the playlist. Finally, a DOS command is built for the opening of the program and the direct play of the songs.

In the second version of the program, instead of using the jlabel object, we implemented a method with which we present the image and draw the circle using the mouse. The two classes to open the image and the data file remain the same. Furthermore, we have implemented three new methods for mouse press, mouse drag and nmouse release. So, the main parts of this version of the program are:

- **mainPanelMousePressed:** This method is used to catch the press of the mouse

- **mainPanelMouseDragged:** This method is used to catch the dragging, which means moving the mouse having the left button pressed.

- **mainPanelMouseDragged:** This method is used to catch the release of the mouse button. After the mouse is released, the playlist is calculated.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Nowadays, with the popularity of multimedia applications, a large amount of music data has been accumulated on the internet. So, automatic classification of music data becomes a critical technique for providing an effective retrieval of music data.

In this work, we have presented an attempt for classification of music into genres. The system was tested, using audio features extracted from segments of wave files. First, it was investigated the usefulness of MFCC-based signal extraction from raw music files. It has shown that MFCC extraction does give information useful for the classification. We have also compared MFCC with other features such the beat and the pitch, and it was discovered that MFCC show an increase in accuracy. The results were promising. In average, 65 percent of the audio data were classified correctly. So, the main conclusion from our results is that it is indeed possible to recognize genre using spectral information.

6.2 Future Work

This approach has raised many interesting questions on which future work could be done. The main drawback of our work it is thought to be the disaggregation of the pieces into genres. The most important in the whole procedure is the training. For that reason, a better organized data set could prove to be crucial for the classification results.

Another idea could be to test and new features, since different features have different impacts on the classifying performance. What is more, we could add and other features, jointly with these that have been presented, for improved classification results. It is probable that a combination of the features would bring better results. Furthermore, another problem of the automatic music genre classification is the design of the classifier. Therefore, as a part of the future work, could be the employment of other classification algorithms or and the fusion with other methods of music classification. Different classification methods could bring better results.

Appendix A

A.1 LAME

Lame is a research project for learning about and improving mp3 encoding technology. It is an open source encoder that can be used to create compressed audio files. These audio files can be played back by every music player. Lame includes an mp3 encoding library, simple frontend application, a much-improved psychoacoustic model and a graphical frame analyzer.

The encoder is a command-line tool, usable from the terminal application or other graphical applications. One of these applications is the lame front-end, which is a graphical interface. It offers easy access to all lame parameters. It can work in batch mode, so it can be used by external applications to encoding/decoding files. However, we chose to use it directly from the terminal, by invoking the lame tool.

The lame offers a lot of options. With the command decode, lame makes the decoding from to a wave file. The input file can be any input type supported by encoding. In our case, it is an mp3 file. What is more, with the command resample sfreq, we achieve the downsampling with the desired sampling frequency. We chose 11 and 16 KHz. What is more, by adding the parameter a at this command, we mix the stereo input file to mono and encode it as mono. The downmix is calculated as the sum of the left and the right channel, attenuated by 6 dB.

Bibliography

[1] Khaled Alsabti, Sanjay Ranka, and Vineet Singh. An efficient k-means clustering algorithm. In *In Proceedings of IPPS/SPDP Workshop on High Performance Data Mining*, 1998.

[2] J.-J. Aucouturier and F. Pachet. Music similarity measures: What's the use? In *Proceedings of 3rd International Conference on Music Information Retrieval*, pages 157–163, Paris, France, 2002.

[3] P. S. Bradley and Usama M. Fayyad. Refining initial points for k-means clustering. pages 91–99. Morgan Kaufmann, 1998.

[4] Hrishikesh Deshp and Rohit Singh. Classification of music signals in the visual domain. In *Proceedings of the Digital Audio Effects Workshop*, 2001.

[5] E. Gomez, A. Klapuri, and B. Meudic. Melody description and extraction in the context of music content processing. *Journal of New Music Research*, 32, 2003.

[6] Mark Levy and Mark Sandler. Lightweight measures for timbral similarity of musical audio. In *AMCMM '06: Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 27–36, New York, NY, USA, 2006.

[7] Tao Li, Mitsunori Ogihara, and Qi Li. A comparative study on content-based music genre classification. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 282–289, New York, NY, USA, 2003.

[8] Stéphane Mallat. *A Wavelet Tour of Signal Processing, Second Edition (Wavelet Analysis & Its Applications)*. Academic Press, September 1999.

[9] Dominc Niewiadonomy and Adam Pelikant. digital speech signal parameterization by mel frequency cepstral coefficients and word boundaries. In *33rd Journal of applied computer science*, 2004.

[10] E Pampalk, A Flexer, and G Widmer. Improvements of audio-based music similarity and genre classificaton. *ISMIR 2005*, 2005.

[11] N. Scaringella, G. Zoia, and D. Mlynek. Automatic genre classification of music content: a survey. *Signal Processing Magazine, IEEE*, 23(2):133–141, 2006.

[12] E. D. Scheirer. Tempo and beat analysis of acoustic musical signals. *Journal of Acoustical Society of America*, 1998.

[13] SOM TOOLBOX official web site. http://www.cis.hut.fi/projects/somtoolbox/.

[14] George Tzanetakis, Georg Essl, and Perry Cook. Musical genre classification of audio signals. In *IEEE Transactions on Speech and Audio Processing*, pages 293–302, 2002.

[15] George Tzanetakis, Ajay Kapur, and Manj Benning. Query-by-beat-boxing: Music retrieval for the dj. In *ISMIR*, 2004.