UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science

Anastasia Semyonova

Music Genre Detection using the Naïve Bayes Classifier

Bachelor's thesis (4 cp)

Supervisor: K. Tretyakov, M.Sc.

Autor: ..... "....." juuni 2009

Juhendaja: ..... "....." juuni 2009

Lubada kaitsmisele

Professor: ..... "....." mai 2009

TARTU 2009

Contents

Introduction

Music in digital form is widely spread nowadays. Many people keep their music libraries on personal computers, mainly in the MP3 format. Various musical compositions are also easily accessible via Internet.

Musical pieces can be grouped into genres according to their sounding characteristics. For most people classification of a given composition is a reasonably easy task. Studies show that listening to 250 milliseconds of sound would often be enough for a human to produce a correct decision [GT07]. Automating this classification process is, however, not so trivial.

Fortunately, we can state the task of digital music classification as a machine learning problem. We consider a set of musical compositions with manually assigned genres as a training set and use it to devise an automatic genre classifier. The traditional approach requires us firstly to extract meaningful features from the acoustic signals, and then apply a general-purpose machine learning algorithm on the transformed data.

For feature extraction we used the ideas proposed in the paper by G. Tzanetakis, G. Essl and P. Cook [GT07]. In their research the authors propose some features that represent music surface and rhythmic structure of audio signals. We reevaluated these features on our own dataset and suggested some additions. Finally, we selected the best performing feature set combining both the original features and our proposed additions.

As long as features are selected properly, the choice of the algorithm is largely irrelevant. In this work we selected the *Naïve Bayes* algorithm due to its conceptual simplicity and efficiency. Our preliminary studies have confirmed that its performance is at least as good as that of some other algorithms, such as SMO, J48, NBTree (Figure 1).

Classification precision of the improved feature set is evaluated by training statistical pattern recognition classifier. Furthermore, our results are compared to those of G. Tzanetakis, G. Essl and P. Cook and found to be very encouraging.

The first chapter of this thesis describes the method of digital representation of music. The next section gives a short overview of machine learning principles and techniques with some detailed information about the Naïve Bayes classifier. The third chapter presents brief descriptions of the features that are used for creating automatic music classification algorithm. Evaluation of the improved algorithm tested on the data set of six music gen-

res (classic, pop, punk, rap/hip-hop, rock and trance) can be found in the fourth chapter. The summary section concludes the results of our research and explains the ideas for future studies. The given paper is supplied with an Appendix A – a CD where code of feature extraction could be found. Appendix B provides more figures to illustrate the results of the work.

Chapter 1

Mathematical Treatment of Music

Before we can approach the problem of automated music classification, we must clarify what music is, and how is it possible to measure similarities between musical compositions.

1.1 Music and Sound

Music is a melodic type of sound. Sound, in turn, is a mechanical disturbance of a medium, either of gas, liquid or solid. For example, when we play the guitar, the movement of the string disturbs the surrounding air, causing the displacement of molecules. Molecules vibrate and transmit this vibration further by striking each other until their initial energy disappears [Ben06].

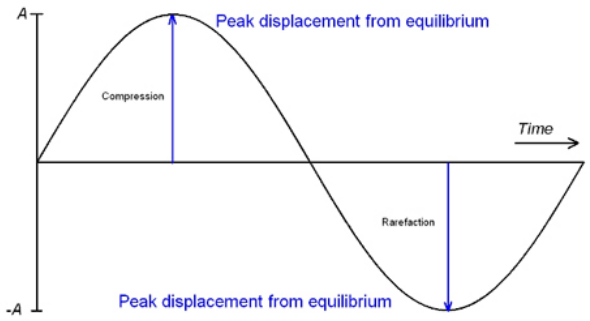


Figure 1.1: The diagram represents the displacement from equilibrium over time creating a sound wave [Jis].

In this manner sound pressure is transmitted from its source to our ears as a wave. In the air sound waves have a pressure which alternately deviates from a state of equilibrium. These deviations are regions of compression and rarefaction of molecules (Figure 1.1). Due to this property sound wave can be represented as a continuous periodic function of time (Figure 1.2).

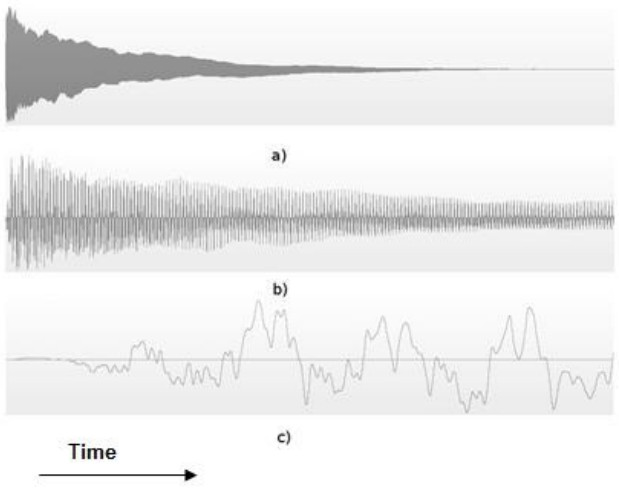


Figure 1.2: Sound wave can be represented as a function of time. The diagrams a), b) and c) show the sound wave in different scales [Jis].

1.2 Digital Representation of Sound

In order to process sound waves on a computer, the corresponding function needs to be converted to a digital form – a process known as sampling. Sampling is performed by measuring the continuous signal at regular intervals (Figure 1.3). Each measurement is referred as a "sample".

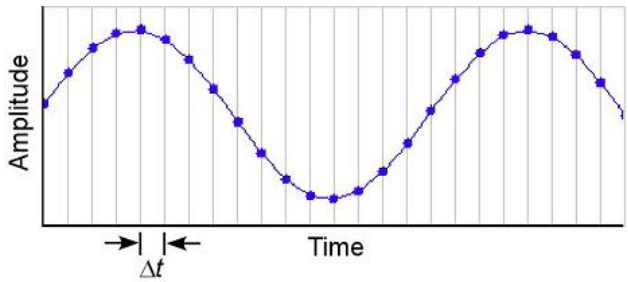


Figure 1.3: The blue sinusoidal curve represents the continuous analog waveform being sampled. Measurements of the instantaneous amplitude of the signal are taken at regular time intervals of length Δt [DR].

Once we have converted the sound wave into a stream of numbers we can store and process this information on a computer. As there are no problems to store numerical information on a machine, that could also do all needed calculations, it seems much more convenient to process sound on a computer.

The data of a music file can be stored in many different ways. One of the main parameters is "sampling frequency" of the stored signal – the number of times per second samples are taken. This attribute, also known as "sampling rate", is measured in Hertz (1 Hz = 1 time per second).

The choice of sampling frequency is an issue we do not deal with in this

work. We only note that in general, the lower the sampling rate of the digitized composition, the more information is lost about the sound wave. From the other side, we should remember the Nyquist-Shannon sampling theorem which says that if a function $x(t)$ contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced $1/(2B)$ seconds apart. Consequently, ideal quality of the record could by achieved with a sampling rate equal to $2B$ (or higher but that is pointless wasting of resources) [Lav04].

The optimal sampling frequency may be different depending on a situation. If we store data for listening, it is always preferable to choose the composition with the highest reasonable ($2B$, where B is maximum frequency perceived by human ear, $\simeq$ 40 kHz) value of sampling rate. For the purposes of analysis, such as automated genre detection, we usually select some golden middle, so that the calculations would be faster and yet no important information is lost.

1.3 Methods of signal analysis

As we know already what music is and how it can be stored in a digital way, it is time to think about the analysis methods that could be applied in a sense of processing numerical representation of music. In our work we use two widely known techniques: spectral analysis and autocorrelation. We describe them briefly in the following sections.

1.3.1 Spectral analysis

One of the classical methods of extracting the properties of a sound wave is known as "Fourier series", "Fourier transform" or sometimes "spectrum". It turns out that any periodic function may be represented as an infinite sum of simpler functions – sine waves (Figure 1.4):

$$S_N f(x) = \frac{a_0}{2} + \sum_{n=1}^{N} [a_n cos(nx) + b_n sin(nx)], N \geq 0$$

for any continuous[1] periodic function f.

The coefficients a_i and b_i of the Fourier series can be found as

$$a_i = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) cos(nt) dt, N \geq 0$$

$$b_i = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) sin(nt) dt, N \geq 1$$

[1] In fact, f need not be strictly continuous, but we avoid these formal details for brevity here.

The decomposition of a continuous function in to the component sine waves is known as the Fourier series, after the 18th century a French mathematician and physicist Joseph Fourier, who discovered it.

As a melody could be represented by the combination of accords also any continuous periodic function, such as sound wave, may be represented as an infinite sum or integral of the simplest waves as sines and cosines (Figure 1.4). An 18th century scientist, named Joseph Fourier (1768 – 1830), proved it mathematically.

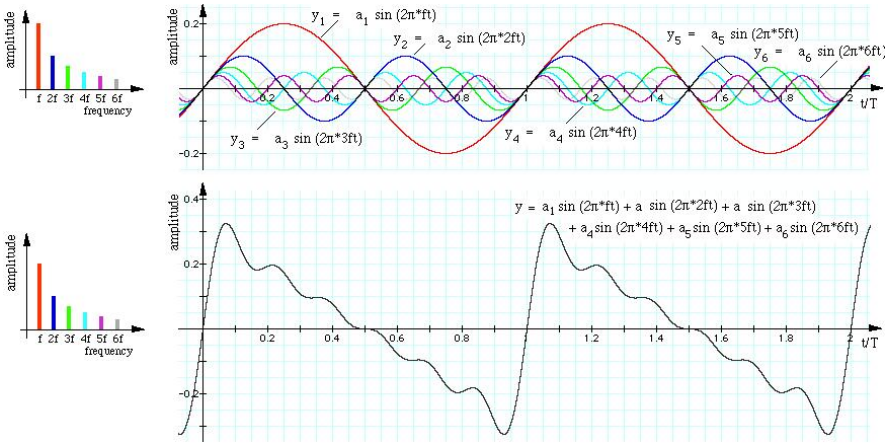


Figure 1.4: Complex sound as a sum of sine waves [DR].

The practical meaning of Fourier transform (FT) is that any sound could be represented as a set of multiple frequencies. Mostly spectrum is used for detecting intensities of those frequencies. The main advantage of it is that our mechanism of sound perception is also based on spectral analysis: in our ears there are special nerves that perceive a vibration of a sound wave, each of them is responsible for a particular frequency being sensitive to it and starting resonate with a larger amplitude of vibration causing an electrical impulse which passes along the auditory nerve towards the brain [Ear]. Therefore, spectral representation is a very valuable technique due to its similarity to the way human perceive sounds.

For music in digital form, a slight modification of the Fourier series is used, known as the "Discrete Fourier Transform (DFT)". The DFT can be computed efficiently using the algorithm known as the Fast Fourier Transform (FFT). We refer the reader to the book of D. Benson [Ben06] for a thorough coverage of these topics.

1.3.2 Autocorrelation

Musical genres often differ a lot in their rhythmical characteristics. To detect the rhythm structure of the composition we use the autocorrelation method.

Autocorrelation is a special case of cross-correlation that, in signal processing, is a measure of similarity of two waveforms as a function of a time-lag

applied to one of them. This is also known as a sliding dot product or inner-product.

The discrete autocorrelation R with lag T is computed as a correlation coefficient between a signal for a discrete signal s_t and a shifted version of it $s_(t - T)$:

$$R_{ss}(T) = \sum_t x_t \bar{x}_{t-T}$$

A high value of autocorrelation coefficient at lag T denotes that the signal s_t and its shifted version $s_(t - T)$ are similar, which could indicate a periodic melodic structure with a period of T samples. Detection of such structure is useful for locating the main beat of the composition.

Chapter 2

Machine Learning Basics

Once a musical composition is represented in terms of numbers, we would like to find an algorithm that assigns a proper genre to each piece of music. Although it might be possible to design such an algorithm "manually", by specifying some common sense rules of thumb this way is too complicated. A much better approach is to collect a data set of labeled musical compositions and build a classifier that generalizes the information in the data. This approach is known as machine learning.

In the following the process of dataset construction and a brief description of the chosen classifier are provided.

2.1 Data

Most classical machine learning techniques require data in the form of vectors. From the previous chapter we already know how to digitize a piece of music and represent it as a vector of numerical values. However, the representation of a musical composition directly as a wave or a spectrum is resource intensive and rather uninformative. A much better approach is to represent sound in terms of a finite set of features – numerical values corresponding to various sounding characteristics valuable for genre classification. Exact specifications of the features are presented in the third chapter.

Finally, when the feature set is fixed, all features are extracted and the data collected, all information could be represented as a matrix with each row corresponding to a piece of music: each column corresponding to a feature and each cell, assigning the value of a given feature, to a given musical composition (Figure 2.1).

2.2 The Choice of Classifier

Suppose that we know exactly that the compositions of one genre are on average longer than the others. 'Classical music" was such a genre in our dataset. In this case, if we were to assign a genre to a musical piece of long

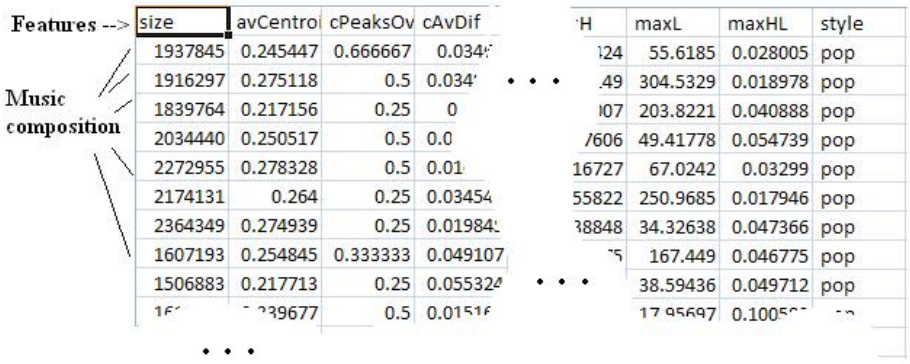


Figure 2.1: In a supervised learning scenario each training example (music composition/file) consists of an object to be classified presented by its features, as well as the correct category (style) to which it should be assigned. All genre names represented by a three-letter acronym.

duration, we would be tempted to classify that piece as belonging to the "classic" genre. This simple idea could be generalized in using probability theory. We have two categories (classes): C (classic) and N (non-classic) genre, and we know the conditional probability distribution of a measured attribute (such as length of the composition) for both classes. In other words, for a given class g we know the probability of obtaining a composition with feature vector x from genre g – $P(x|g)$. Usually we know something about these distributions (as in example above, we know that the probability of comparably long composition of the category C is one or 100%). What we want to know is what is the most probable genre of a given a composition x ($P(g|x)$) [Tre04] that due to the well known in probability theory Bayes theorem could be calculated as follows:

$$P(g|x) = \frac{P(x|g)P(g)}{P(x)} = \frac{P(x|g)P(g)}{P(x|N)P(N) + P(x|C)P(C)}$$

where P(x) denotes the a-priori probability of composition x, and P(g) – the a-priori probability of class g (i.e. the probability that a random composition is of that genre). So if we know the values $P(g)$ and $P(x|g)$ (for $G = \{C, N\}$), we may determine $P(g|x)$, which is already a nice achievement that allows us to use the following classification rule: If $P(C|x) > P(N|x)$, classify x as classic, otherwise classify it as non-classic. This is the so-called maximum a-posteriori probability (MAP) rule. Using the Bayes formula we can transform it to the form: If

$$\frac{P(x|S)}{P(x|L)} > \frac{P(L)}{P(S)}$$

classify x as classic, otherwise classify it as non-classic composition.

Of course, our example is very simple: usually it is not possible to find a feature that could guarantee 100% probability of some class, so we are

forced to use a set of features for obtaining much more precise classification result; also the number of categories could be much bigger. In those cases the decision making process becomes complicated. To simplify them some algorithms were built, one of them, Naïve Bayes, was chosen to solve our problem of automatic genre classification.

2.2.1 The Naïve Bayes Classifier

In the present paper we selected the *Naïve Bayes* algorithm for automatic genre classification mostly due to its conceptual simplicity and comparably good efficiency. Our preliminary studies have confirmed that its performance is not worse than other algorithms, such as SMO, J48, NBTree (Figure 1).

The Naïve Bayes classifier is a classification method that is used for categorical data based on applying Bayes' theorem. By the classical Bayes approach, for a record to be classified, the categories of the predictor variables are noted and the record is classified according to the most frequent class among the same values of those predictor variables in the training set. A rigorous application of the Bayes theorem would require availability of all possible combinations of the values of the predictor variables:

$$p(G, F_1, F_2, \ldots, F_n) = p(G)p(F_1, F_2, \ldots, F_n|G) =$$
$$= p(G)p(F_1|G)p(F_2, \ldots, F_n|G, F_1) =$$
$$= p(G)p(F1|G)p(F_2|G, F_1)p(F_3|G, F_1, F_2)\ldots p(F_n|C, F_1, F_2, \ldots F_n - 1)$$

When the number of variables is large enough, this requires a training set of unrealistically large size.

The Naïve Bayes method overcomes this practical limitation of the rigorous Bayes approach to classification. The major idea of it is to use the assumption that predictor variables are independent random variables. This makes it possible to compute probabilities required by the Bayes formula from a relatively small training set:

$$p(G, F_1, F_2, \ldots, F_n) \approx p(G)p(F_1|G)p(F_2|G)\ldots p(F_n|G) = p(G)p(F_i|G)$$

So, we Naïve Bayes assumption could be presented by formula:

$$p(G|F1, F2, \ldots Fn) \approx \frac{1}{Z}p(G)\prod_{i=1}^{n} p(F_i|G)$$

where Z is a scaling factor dependent only on $F_1, F_2, \ldots, F_n$, i.e., a constant if the values of the feature variables are known.

To sum up, we can say that in spite of its naive design and over-simplified assumption, Naïve Bayes classifiers often work much better in many complex real-world situations than one might expect. An advantage of the Naïve Bayes classifier is that it requires a relatively small amount of training data to estimate the parameters necessary for classification.

2.3 Data Validation

To validate the performance of our algorithm we used the 10-fold cross validation technique. The idea of the approach is to split the training set into 10 parts, use 9 for training and 1 for testing (measuring the precision), and then repeating this procedure for each of the 10 parts and taking the average of the 10 obtained precision estimates. This validation method is known to produce reasonably accurate results even when the training set is small [IHW05].

Chapter 3

Algorithm for Music Classification

3.1 Statement of the Problem

What we ultimately wish to obtain is an automatic musical genre classification algorithm, that is a decision function that would tell us with what probability a music composition is a representative of one of six defined genres: classic, pop, punk, rap/hip-hop, rock and trance. We shall look for this function by training the Naïve Bayes algorithm on a set of manually pre-classified music compositions. This is nearly a general statement of the standard machine learning problem. Therefore, the only thing we need is a set of meaningful features

3.2 Signal Features for Genre Classification

The choice of informative sound features is crucial for the performance of the genre classification algorithm. Genre classification is an artificial division of musical compositions into several groups made by people, so it is important to understand how people perceive music and how sounding characteristics could be represented in a numerical way understandable for a computer. As is customary, the psychological characteristics of music could be classified into four groups: tonal, dynamic, temporal and qualitative [Ols67]. The paper that we base our work on [GT07] proposes 17 acoustical features that describe music surface(tonal) and rhythm (temporal) aspects of sounding characteristics. The features of the first category corresponds to texture, timbre and instrumentation of the composition. The features of the second served to describe rhythmic structure of music: time, duration, tempo, rhythm.

We have implemented extraction of the features proposed in the paper and applied to our test data which contained somewhat different set of genres. As the results were less than optimal, we decided to explore some of our own ideas to improve classification accuracy.

This section provides descriptions for all the features used in the experiments that follow. In the fourth chapter we evaluate our proposed features and assess the degree of improvement.

3.2.1 Data processing

Our data is a collection of 300 MP3 files containing 50 music compositions of each of the following genres: classic, pop, punk, rap/hip-hop, rock, trance.

For analyzing sound waves, that we received by presenting music files in a digital format, we used Weka – a powerful suite of machine learning software which supports several standard data mining tasks, such as clustering, classification, visualization and feature selection that we have used in our work.

To process data and convert it to an appropriate format accepted by Weka, we used Scilab [Sci] – an open source software for mathematical calculations, that supports reading and writing of sound files in WAV format. As Scilab deals only with sound files in the WAV format. Finally, we used Sound eXchange [Sox] – a sound converter software to convert MP3 files to the WAV.

To reduce the size of data and save on computation, we decreased the sample rate of all compositions from 44 kHz down to 8 kHz. Manual inspection showed that this transformation still conserved enough signal to be able to recognize the genre by ear. We use the Comma Separated Values (CSV) format to hold the intermediate data tables where all the information except of feature and genre names is numeric and presented in a following way (Figure 2.1).

3.2.2 Texture and Analysis Windows

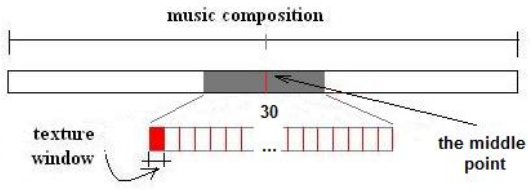


Figure 3.1: Feature extraction: location of texture windows.

Following the methodology proposed in the study [GT07], to classify the whole composition we use a short fragment of length 1 second (= 8000 samples). We refer to this fragment as the *texture window*. Together there are 30 texture windows for each composition situated as far as possible close to the middle of composition (Figure 3.1) because we hope sounding there is the most typical for the given genre. Each texture window is further split

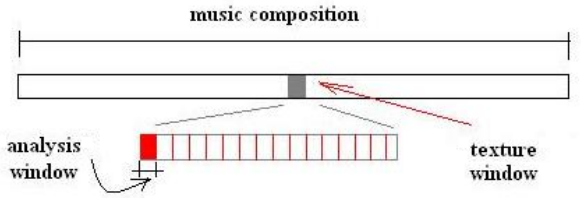


Figure 3.2: Feature extraction: texture and analysis windows

into 15 smaller windows of equal length – *analysis windows* each of them 512 samples long (Figure 3.2). As a result, in total from each data point we had 30 records and from the whole data set 6 styles*50 files*30 instances = 9000 instances for further analysis.

For classification we used the features computed from the texture window. We name our features using short codes such as `lowEnergy` to be able to refer to them conveniently and mark the features that we propose with an asterisk (e.g. `size*`).

All features, except for `size*` and `lowEnergy`, are computed as the mean and standard deviation of the corresponding metrics of the analysis window. Therefore, for each analysis window metric (such as, `wavZero`), we defined two features to be used in the machine learning algorithm: average (`_mean`) and standard deviation (`_std`) (e.g. `wavZero_mean` and `wavZero_std`).

3.2.3 The Features

All features presented in this section proposed by us (marked with an asteriks) or by the authors of the paper [GT07]. As mentioned before, we consider two types of features: music surface and rhythm features. Music surface features can be further grouped into two categories – those that are based on the time-domain characteristics of the signal ("wave features") and those that are based on the spectral representation ("analysis of frequencies" aka "spectral features").

3.2.4 Wave Features

In this section we present features obtained from a pure audio signal wave.

Low energy (`lowEnergy`)

One of the ideas proposed in the paper is to measure how the level of loudness is changing over the composition. As we know from our own experience, songs of rap/hip-hop genre have the same level of loudness all the time but we can not say the same about pop compositions where jumps and falls of loudness

are more common. This can be easily seen from the plot of `lowEnergy_mean` feature values (Figure 4).

`LowEnergy` is the percentage of analysis windows that have energy (and therefore loudness) less than average energy of the analysis windows over the texture window.

Energy for one analysis window i is calculated by formula:

$$energy_i = \sum_{j=1}^{512} {w_j}^2.$$

Zero crossings (`wavZero`)

Authors of the paper mentioned that compositions of different genres have different amount of noise in their signals. To measure it they propose to count the number of zero crossing of the signal within the analysis window. We denote this feature as `wavZero`.

As we see from the plot (Figure 5), the amount of the noise in classic music differs a lot from any other genre compositions, especially from trance and rap/hip-hop ones, what intuitively can be the truth.

Average amplitude (`wavAvAmpl*`)

A particularly useful feature can be obtained by measuring the difference $wavAvAmpl_i$ between the highest and the lowest peaks of the signal within each analysis window w_i as follows:

$$wavAvAmpl_i = \max(w_i) - \min(w_i).$$

Mean amplitude (`wavAvAmpl_mean*`) is often higher for louder compositions. This could, for example, help to discriminate punk from classical music (see Figure 6). Standard deviation of amplitudes (`wavAvAmpl_std*`, in turn, can detect compositions with a big and small difference in loudness of the signal of different analysis windows, such as rap/hip-hop and classic (Figure 7).

Average difference between sample values (`wavAvDif*`)

We obtain the value of the `WavAvDif` feature by measuring the rate of the signal level change within each neighbor sample values. After that we find the average of that value for each analysis window.

For $wavDif_j = |w_{j+1} - w_j|$ – difference between neighbor sample values

$$wavAvDif_i = \frac{(\sum_{j=1}^{512} wavDif_j)}{512}$$

is average difference between sample values on the analysis window i.

It is worth to say that two main characteristics of sound, such as loudness and main frequency, are constantly dependent on the energy of signal. Therefore, `WavAvDif`, that explores monotony of the signal, helps to distinguish loud compositions of high frequencies (maximum values of the feature) from opposite ones. `WavAvDif` shows whether the signal changes a lot over a texture window or its value is more or less constant. As we see from the plot (Figure 8) the signal of a rap/hip-hop music could be named monotonous (quiet and bass) in comparison with a punk one.

Length of the composition (`size*`)

One of the most simple but still effective properties is the length of a composition measured in seconds. This feature stands somewhat separate from the other ones here as it does not use windows and requires the whole composition to be available. However, it provides a significant classification improvement: as illustrated by the plot (in Figure 9), classic and trance compositions are usually much longer then pop and punk ones.

3.2.5 Analysis of frequencies

Another approach of music analysis is based on a spectral characteristic (Fourier Transform) of a sound signal that leads us to the understanding of a music as a sum of signals of different frequencies. Knowing the frequencies that exist in our signal we can calculate average frequency of it. For easier calculations we are going to use Fast Fourier Transform (FFT) – an efficient algorithm to compute the discrete Fourier transform [CW65]. All features represented in this chapter are based on the value of an average frequency ($centroid_i$) of a signal over analysis window calculated by following formula:

$$centroid_i = \frac{\sum_{i=1}^{512} f_i M[f_i]}{\sum_{i=1}^{512} M[f_i]}.$$

where $M[f_i]$ is the magnitude of the FFT at frequency bin f_i within 512 bins of the analysis window.

Centroid (`centroid`)

The simplest way to analyze the role of an average frequency of an audio signal is to measure its mean (`centroid_mean`) and standard deviation(`centroid_std`) values over texture window.

From the plot (Figure 10) we can see that the minimal average frequency over texture window (`centroid_mean`) is detected in Trance compositions and maximum in Rock ones. Furthermore, `centroid_std` feature helps us distinguish classic music that includes mostly one type of frequencies from the others, where frequencies vary greatly (Figure 11).

Average change of average frequency (`flux`)

Having the value of the average frequency of the signal for each analysis window w_i we can think about measuring how it differs from one analysis window to another by computing the difference by the following formula:

$$flux_i = |centroid_{i+1} - centroid_i|$$

.

As our experiments showed (Figure 12), the mean difference between neighbor analysis windows over the whole texture window (`flux_mean`) helps to detect very impulsive compositions that have frequent change of frequency, such as in rap/hip-hop songs, in comparison to classical music.

The Maximum High and Low Frequency Intensity (`maxH*`, `maxL*`)

Another powerful feature based on frequency analysis is a value of the maximum intensity of high (`maxH*`) and low (`maxL*`) frequencies. Usually, for example, classic music may have `maxL_mean*` value much smaller then punk music because accent is made for bass in the last one (Figure 14). Standard deviation of maximums of high frequencies (`maxH_std*`) detects how big is range of high frequencies in the composition: for classic and rap/hip-hop it's quite small, for punk – vice versa – range of high frequencies is really big (Figure 13).

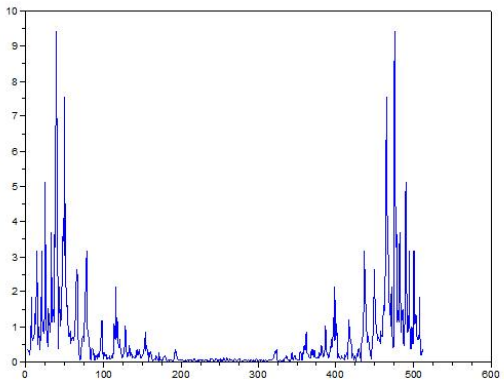


Figure 3.3: Frequency intensities of the analysis window of the signal are center-symmetrical

To find the value of `maxH*` (`maxL*`) we find the maximum of high (low) frequency intensity for each analysis window simply by comparing the values received after FFT. As the values of FFT are symmetrical (Figure 3.3, so we can explore only half of them, concerning $s_i = (s_1, \ldots, s_{256})$ as a representation of a frequency-domain for the analysis window i where the first half of

the values $(s_1, \ldots, s_{128})$ give us overview over low-frequencies intensity, and, correspondingly, $(s_{129}, \ldots, s_{256})$ represents intensity of high-frequencies.

3.2.6 Rhythm Features

In this section we present features that describe rhythmic characteristic of music. Ideas of all of them are taken from the paper but calculations are simplified.

To compute these features we do not use the texture/analysis windows framework as before. Instead, we select a fragment of length 2 seconds, subsample the signal to 1000 Hz, compute the autocorrelation of the resulting vector and detect five highest peaks of the autocorrelation function.

For detecting five highest peaks we use following strategy: on each and every step we find the maximum value of the ordinate vector and control whether values of its 50 neighbors from both sides are lower. If not we change the values of the controlled point and 10 his neighbors to the average value and continue our search until we find all peaks we need.

Relative Amplitudes (`a0`, `a1`)

The feature `a0` value is calculated as an relative amplitude (divided by sum of all five amplitudes) of the first peak; `a1` – of the second peak. Both of them mainly help to separate trance music compositions from others (Figure 15, 16).

3.2.7 The Feature Vector

Altogether we end up with a feature vector consisting of 18 elements. To clarify whether all of them are really needed we used the a feature set evaluation facilities of Weka (ClassifierSubsetEval, greedy search) to find the optimal feature set. As a result we obtained the following 13-dimensional feature vector: (`a0`, `a1`, `centroid_mean`, `centroid_std`, `flux_mean`, `lowEnergy`, `maxH_std*`, `maxL_mean*`, `wavZero_std`, `wavAvAmpl_mean*`, `wavAvAmpl_std*`, `wavAvDif_mean*`, `size*`).

As we see from the table (Figure 3) the final feature set is a combination of 5 features from the original paper, 2 simplified features from the paper and 6 our proposals of meaningful features. Furthermore, it includes features of two types: music surface(11) and rhythm(2) features. Features in the final feature set represent also both calculation bases: wave(8) and spectral(5) features.

Chapter 4

Evaluation of Results

We applied the feature extraction and classification techniques, described in the previous chapters, on our data set. This chapter introduces evaluation of the obtained results.

To illustrate the usefulness of each separate feature, we shall illustrate its ability to discriminate among six genres (classic, pop, punk, rap/hip-hop, rock and trance) that we used in our experiments.

4.1 Classification Precision

To observe the classification results we used the confusion matrix – a matrix where each column represents the instances of a predicted genre, while each row represents the instances of an actual genre. One benefit of a confusion matrix is that it is easy to see if the system is confusing two genres.

For evaluation of efficiency of proposed algorithm we measured classification precision for two different tasks. One of them is the ability to distinguish compositions of a given genre from all others. Another task is to detect which of those six genres suits the best for a given composition. The results are presented below.

4.1.1 One Genre vs Others Classification results

To evaluate our features we wanted firstly to know how successfully compositions of a given genre are distinguished from the whole data set. For this purpose we used data labeled for two genres: one genre and all other compositions. On average more than 77% of compositions were classified correctly, and in the case of classical music the classification precision got up to 94.8%, which we consider to be a rather successful result. Figure 4.1 provides more information about one genre classification.

The following plot (Figure 4.2), made of the data proposed by the confusion matrix for two classes (`genre` and `other`), introduces the percentage of misclassification for detecting `genre` compositions from `other` (genre mis-

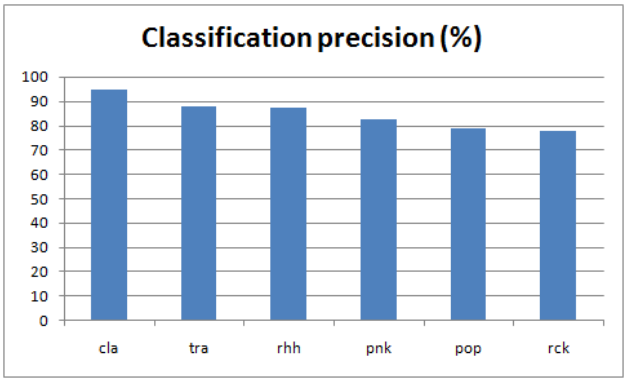


Figure 4.1: Evaluation of correctly detecting each genre from others.

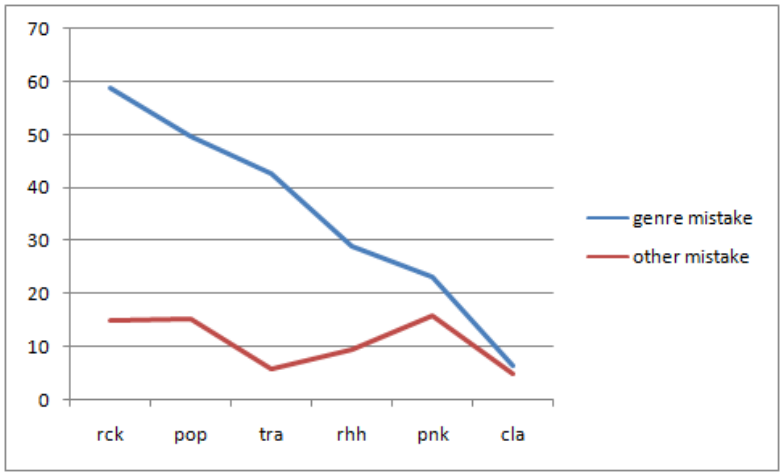


Figure 4.2: Percentage of wrongly classified compositions between `genre` and `others`.

take) and for distinguishing entities of other genres from `genre` compositions (other mistake).

As we see from the plot, both mistakes are not equal by their value. For all classes genre mistake is higher than other mistake but there are genres for which this difference is abnormally big. The most unreliable is distinguishing rock compositions – 59% , pop and trance entities are also classified with less than 60% correctness. As the results of our research show that distinguishing compositions of classic genre from others is the easiest task for our algorithm. According to distinguishing other compositions from `genre` ones: in all cases mistake percentage is not bigger than 16%.

4.1.2 All Genres Classification results

The results of all genre classification are significantly better than the same by random classification (Figure 2). Classification precision of our algorithm (61,6%) for our music genre set is even higher than of algorithm proposed

by the authors of the paper which classifies about 57,5% of the compositions correctly.

```
=== Confusion Matrix ===

   a    b    c    d    e    f   <-- classified as
 586  171   79  376  160  128 |    a =  pop
 170  618  146  350  125   91 |    b =  rck
   3   83 1380    3    9   22 |    c =  cla
 142  138    5 1091   49   75 |    d =  pnk
 268   65   24   92 1008   43 |    e =  rhh
 199   82   39  125  193  862 |    f =  tra
```

The observation of the confusion matrix (Figure ??) indicates that pop and rock genres are most problematic for classification, these two genres being often confused for punk music. There are several reasons for the misclassification. Firstly, quite often a genre assignment to a musical composition is strongly biased by non-acoustic factors, such as the brand and image of the performer. For example, a number of compositions considered as punk and hip-hop are difficult to discern from pop to a non-specialist. Secondly it is quite popular nowadays to mix genres, for example, adding a violin part in some single parts of the composition. There could be even some special mixed-genres but in our work we accept only six really important music types. And finally, our choice of the features may still not be optimal to classify the chosen set of genres.

4.2 Comparing the Results

Comparing our results to those from the paper we based our work on, first of all, we should note that data set was very different in both cases. This could be a reason why most of the features used in the paper gave worse results in our case. That was the reason why we decided to propose our own features that improved the algorithm of music genre classification. But we should acknowledge that result is still far from the ideal.

Summary

In this thesis we presented a study of an approach to automated musical genre classification. Besides a brief overview of the theoretical background, we documented our approach, the experiments we performed and the results we obtained.

In the practical part of our work we extracted features proposed in the paper [GT07] and evaluated their work on our data set using different algorithms. Afterwards, we fixed the classifier, made some proposals for improving the feature set, extracted new features and evaluated their classification precision. Finally, we constructed new feature set of 13 elements that classifies music of six genres with the accuracy of 61,6% that is almost four times better than random.

There are several directions for future research. One of the obvious is to continue work on the improvement of musical genre classification algorithms, searching for new valuable features that could be extracted from the audio signal. From the other point of view, we could think about another application of already known methods of sound wave processing. For example, it is natural for human beings to describe music by some artificial characteristics, such as "positive", "negative", "aggressive", even more complicated like music for driving, relaxing, or even typical music for horror movies. However, there is no stable connection between music genres and such characteristics. This is the reason why we could try to find an algorithm for music classification by another categorical descriptions – "artificial genres". This could provide an easier search for a particular set of music compositions that will help people to construct their play-lists automatically depending on their mood, desire or context. Moreover, we believe that such an algorithm could simplify the work of professional sound-producers; they will be able to get intelligent proposals of music compositions by their artificial characteristics.

Muusika žanri avastamine kasutades Naïve Bayes klassifikaatori.

Bakalaureusetöö (4ap)

Anastassia Semjonova

Resümee

Tänapäeval hoitakse muusikat peamiselt digitaalvormis MP3 formaadis nii arvutis kui ka internetis. Muusika faile on nii palju, et neid tuleb kuidagi klassifitseerida. Kompositsioone on võimalik grupeerida žanrideks heliseva iseloomu järgi. Paljud inimesed, nagu uuringud näitasid, oskavad muusika fragmendi klassifitseerida lühikese (250 millisekundi) kuulamisaja jooksul, kuigi selle protsessi automatiseerimine on palju keerulisem ülesanne. Kaesoleva töö eesmargiks on lahendada see problem, kasutades masinõpe meetodeid.

Töö baseerub G. Tzanetakis, G. Essl ja P. Cooki artiklil [GT07], mis käsitleb muusika žanri avastamisalgoritmi loomise temaatikat. Peamiseks ideeks on esitada muusika faile numbrilises formaadis ja võta välja sellest informatsioonist mõned tunnused, mis kirjeldaksid muusika helisust ja aitaks žanrideks klassifitseerimisel. Esiteks, me realiseerisime artiklis pakutud tunnuste arvutamismeetodit ja hinnasime nende töö meie andmestikul, mis koosneb 300 muusika failidest – iga žanri (klassika, pop, punk, rap/hip-hop, rokk ja trance) esitavad 50 kompositsiooni. Seejärel valisime klassifikaatorit ja pakkusime välja oma ideed. Tulemuseks me saime 13-elemendilist tunnuste vektori, mis pooleli koosneb baseeruvas artiklis esitatud tunnustest ja pooleli meie ideedest. Tunnuste vektor koos valitud algoritmiga võimaldavad klassifitseerida 6 žanri lood 61,6% täpsusega, mis on peaaegu neli korda parem kui juhuslik. Peale seda, tulemus on 5% parem kui see mida said baseeruva artikli autorid.

Käesolevas töös püstitatud probleem on väga aktuaalne tänapäeval ja seal on palju suundi edasisteks uuringuteks. Näiteks, võib uurida, kuidas võib muusika žanri klassifikaatori paremaks teha. Samal ajal võib tegeleda teiste klassifikaatorite loomisega, mis baseeruvad teistel klassifitseerimisideedel. Päris huvitav ja kasulik võiks olla algoritm, mis eraldab positiivse ja negatiivse varjundiga kompositsioone.

References

[Ben06] Dave Benson. *Music: A Mathematical Offering.* 2006.

[CW65] James W. Cooley and John W. "an algorithm for the machine calculation of complex fourier series" (fft - fast fourier transform). 1965.

[DR] Digital representation of sound. http://www.birds.cornell.edu/brp/pdf-documents/AppA-DigitalSound.pdf.

[Ear] Physics tutorial. http://www.ddart.net/science/physics/.

[GT07] P. Cook G. Tzanetakis, G. Essl. Automatic music genre classification of audio signals. *EURASIP Journal on Applied Signal Processing*, 2007.

[IHW05] Eibe Frank Ian H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition).* 2005.

[Jis] Jisc – digital media helpdesk. http://www.jiscdigitalmedia.ac.uk/audio/advice/the-physical-principles-of-sound/.

[Lav04] Dan Lavry. Sampling theory. urlhttp://www.lavryengineering.com. 2004.

[Ols67] Harry F. Olson. *Music, physics and engineering.* 1967.

[Sci] Scilab – the computer algebra system. http://www.scilab.org/.

[Sox] Sox – sound converter software. http://sox.sourceforge.net/.

[Tre04] Konstantin Tretyakov. Machine learning techniques in spam filtering. 2004.

Appendices

Appendix A: Program code (on a compact disc)
Appendix B: Figures

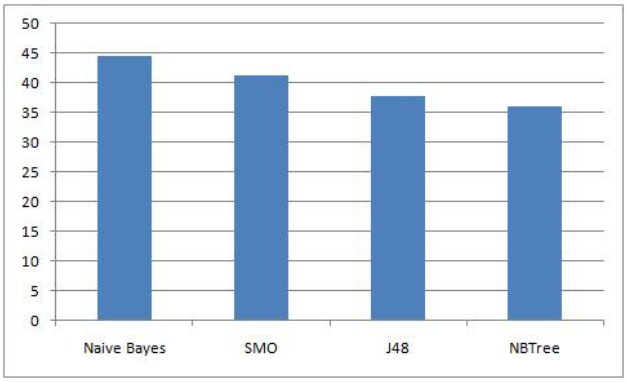


Figure 1: Percentage of correctly classified compositions obtained by 10-fold evaluation strategy on the dataset of six genres (classic, pop, punk, rap/hip-hop, rock, trance) using different algorithms. The feature set contains features of music surface and rhythm proposed in the paper [GT07].

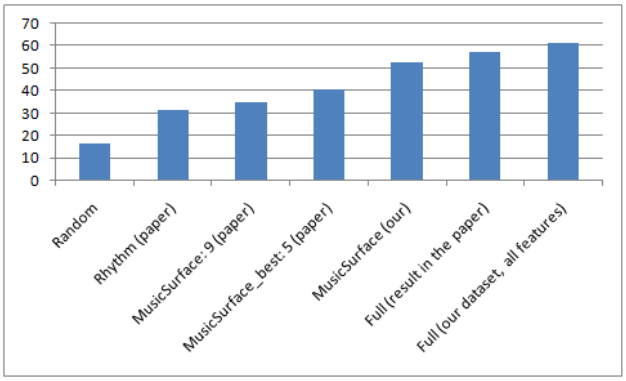


Figure 2: Relative feature set importance. These classification results are calculated using a 10-fold evaluation model.

feature code	feature source	basis of calculations	feature type	included
a0	old, modified	spectral	rhythm	yes
a1	old, modified	spectral	rhythm	yes
centroid_mean	old	spectral	surface	yes
centroid_std	old	spectral	surface	yes
flux_mean	old	spectral	surface	yes
flux_std	old	spectral	surface	no
maxH_mean	new	spectral	surface	no
maxH_std	new	spectral	surface	yes
maxL_mean	new	spectral	surface	yes
maxL_std	new	spectral	surface	no
lowEnergy	old	wave	surface	yes
size	new	wave	surface	yes
wavAvAmpl_mean	new	wave	surface	yes
wavAvAmpl_std	new	wave	surface	yes
wavAvDif_mean	new	wave	surface	yes
wavAvDif_std	new	wave	surface	no
wavZero_mean	old	wave	surface	no
wavZero_std	old	wave	surface	yes

Figure 3: Summary table of the features tested.

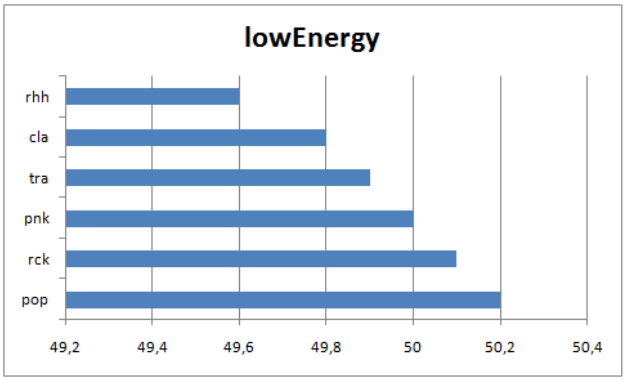


Figure 4: `lowEnergy` feature value for the compositions of different genres

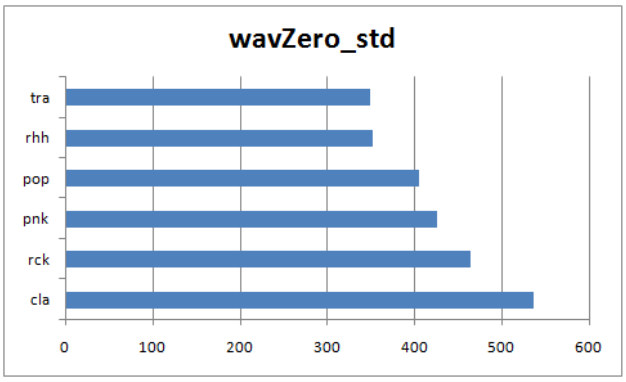


Figure 5: `wavZero_std` feature value for the compositions of different genres

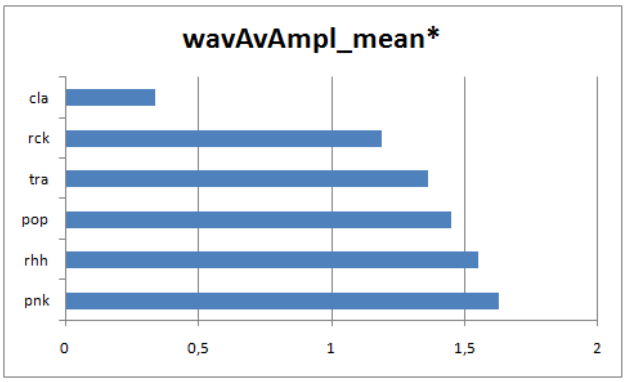


Figure 6: `wavAvAmpl_mean*` feature value for the compositions of different genres

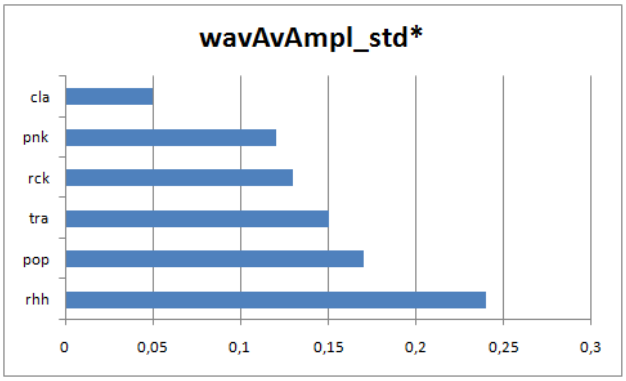


Figure 7: `wavAvAmpl_std*` feature value for the compositions of different genres

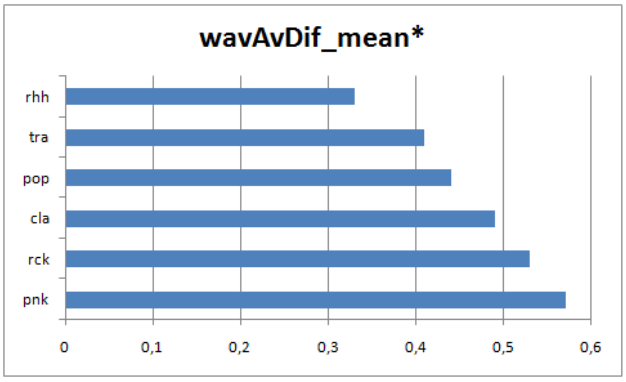


Figure 8: `wavAvDif_mean*` feature value for the compositions of different genres

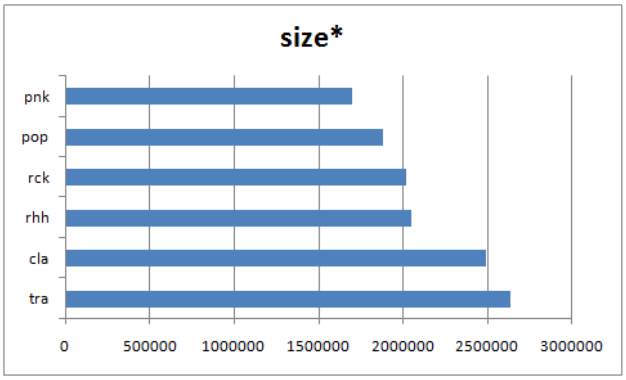


Figure 9: Average length of the compositions of different genres

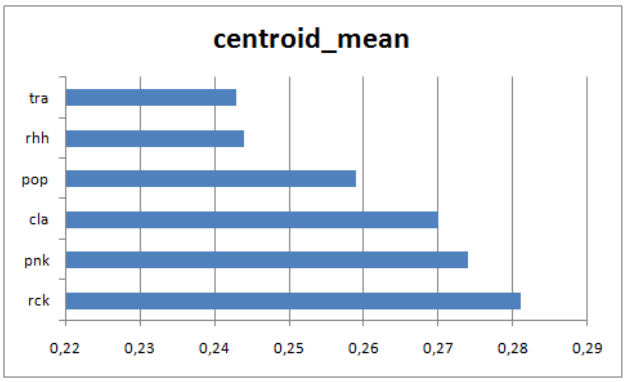


Figure 10: `centroid_mean` feature value for the compositions of different genres

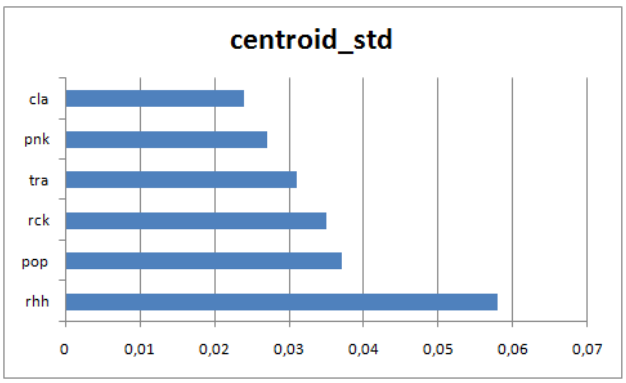


Figure 11: `centroid_std` feature value for the compositions of different genres

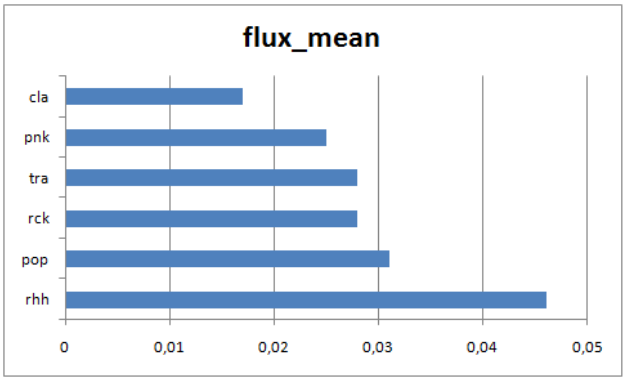


Figure 12: `flux_mean` feature value for the compositions of different genres

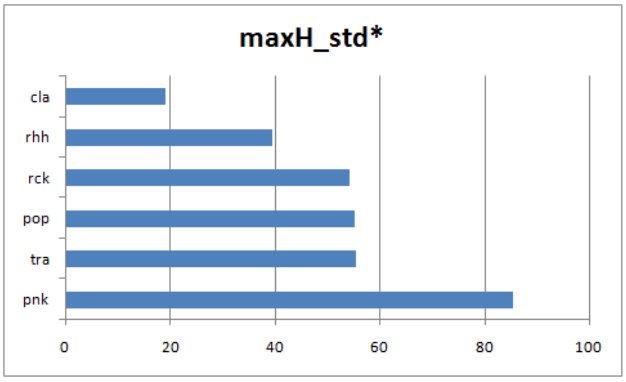


Figure 13: `maxH_std*` feature value for the compositions of different genres

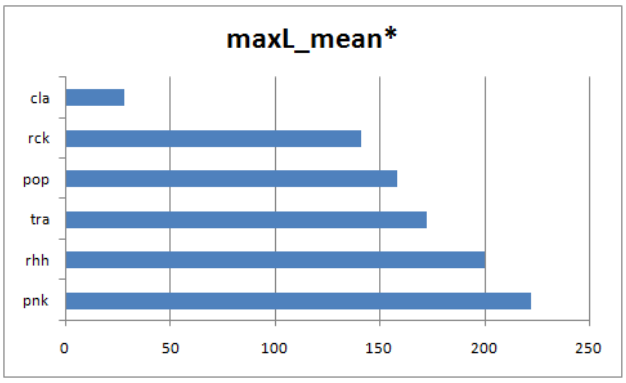


Figure 14: `maxL_mean*` feature value for the compositions of different genres

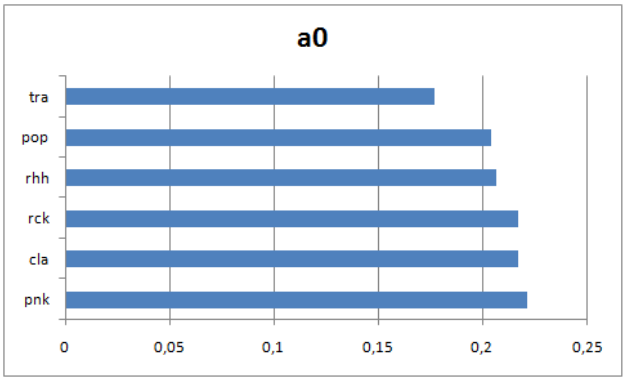


Figure 15: `a0` feature value for the compositions of different genres

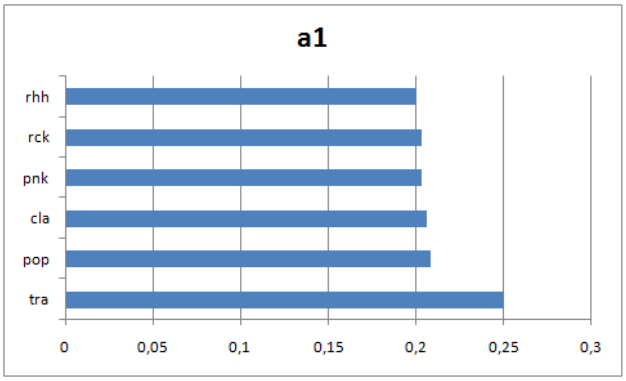
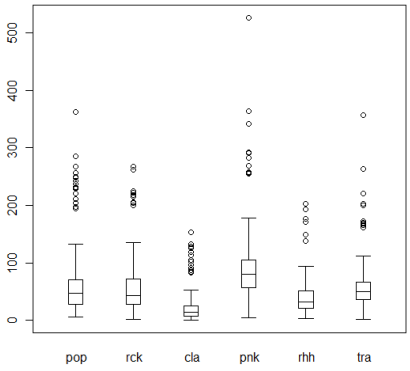
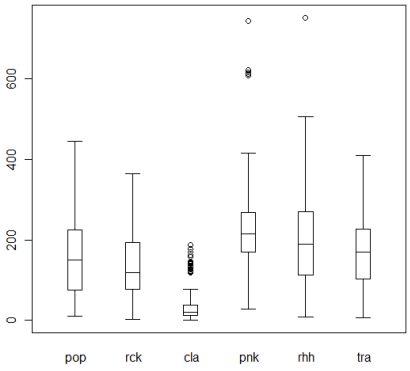


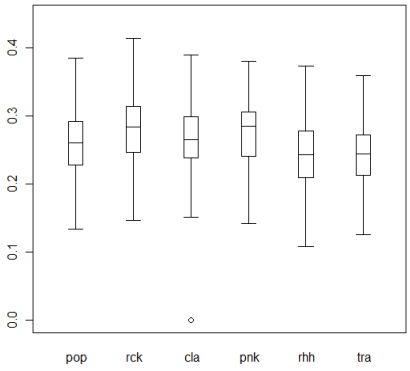
Figure 16: `a1` feature value for the compositions of different genres



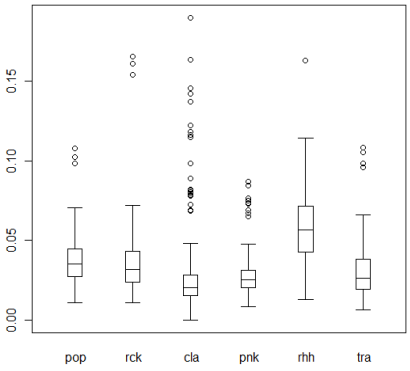
(a) maxH_std*



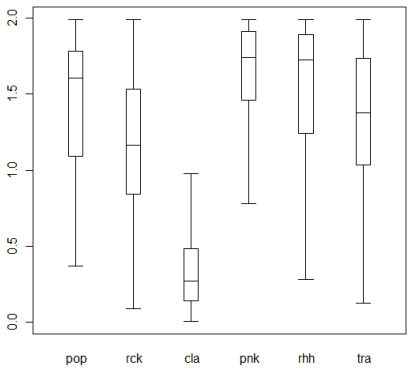
(b) maxL_mean*



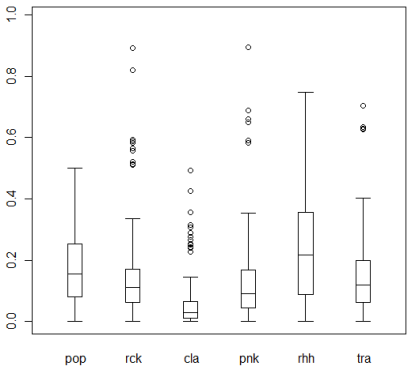
(c) centroid_mean



(d) centroid_std

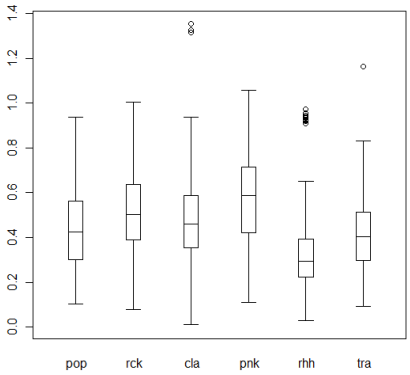


(e) wavAvAmpl_mean*

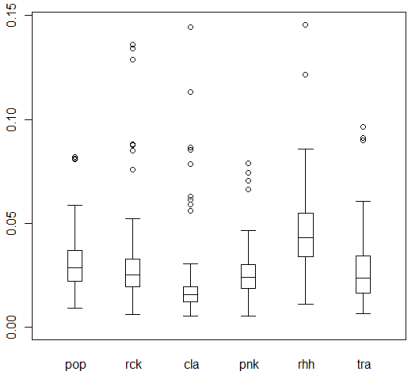


(f) wavAvAmpl_std*

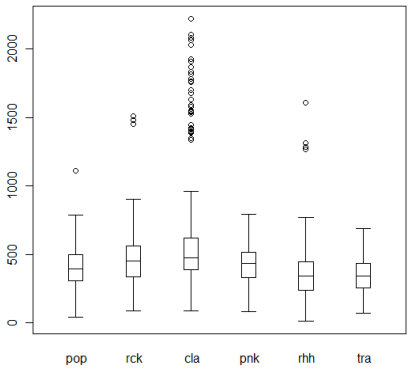
Figure 17: Boxplot of the features' values (part 1)



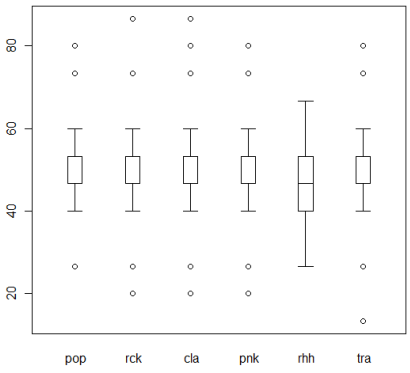
(a) wavAvDif_mean*



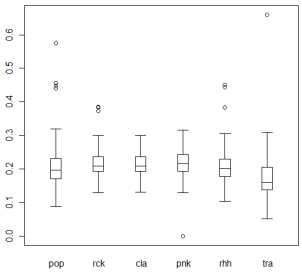
(b) flux_mean



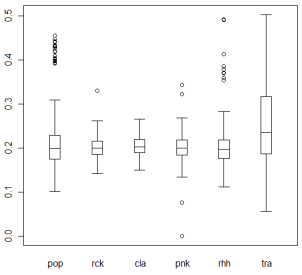
(c) wavZero_std



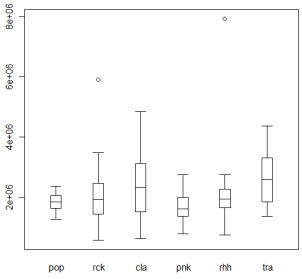
(d) lowEnergy



(e) a0



(f) a1



(g) size*

Figure 18: Boxplot of the features' values (part 2)