UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ALAN FRANCISCO HÖNG

A Machine Learning Approach for Content-Based Music Recommender Systems

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Paulo Martins Engel

Porto Alegre
December 2015

“Computers are the equivalent of a bicycles for our minds”

— STEVE JOBS

ABSTRACT

This work tries to approach music recommendation in a content based way. Most today‘s recommender systems use user data on such as listening history and likes to recommend new music to users. Many of todays music platforms such as Soundcloud, Mixcloud or Youtube grow rapidly and contain many new music pieces of lesser known artists. As traditional recommender systems require a great amount of user data to predict good recommendations, these approaches might perform poorly on these newer platforms, as lesser known tracks won’t be recommended at all or only appear to very few users. This work presents a recommender system which requires very few user data to make recommendations. By analysing audio data and clustering the analysed data using a neural network, recommendations for each user are generated. At the end we evaluate the achieved results using offline evaluation techniques.

Keywords: Music recommendation. self-organising map. music similarity. audio analysis.

Uma abordagem em aprendizagem de máquina para sistemas de recomendações de musica baseados em conteúdo

RESUMO

Este trabalho trata de abordar recomendações musicais de uma maneira baseada em conteúdo. A maioria dos sistemas de recomendação hoje em dia usam os dados do usuário como o histórico de musicas escutadas e preferencias para recomendar novas músicas aos usuários. Muitas das plataformas recentes de música tais como Soundcloud, Mixcloud ou Youtube crescem rapidamente e contêm muitas novas peças de música de artistas menos conhecidos. Como sistemas de recomendação tradicionais requerem uma grande quantidade de dados de usuários para prever recomendações, essas abordagens podem executar mal nessas plataformas mais recentes, como musicas menos conhecidas não será recomendada a todos ou só apareceram para poucos usuários. Este trabalho apresenta um sistema de recomendações que usa um mínimo de dados de usuário para fazer recomendações. Ao analisar os dados de áudio e agrupando-os, utilizando uma rede neural, as recomendações para cada usuário são gerados. No final serão avaliados os resultados obtidos através da avaliação de técnicas off-line.

Palavras-chave: Reccomendações de musica, self-organising map, semelhança de musica, analise de audio.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

SOM	Self Organizing Map
IGMN	Incremental Gaussian Mixture Network
DTFT	Discrete Time Fourier Transform
STFT	Short Time Fourier Transform
MIR	Music Information Retrieval
MSD	Million Song Database
FFT	Fast Fourier Transform
SPL	Sound Pressure Level
CF	Collaborative Filtering
GMM	Gaussian Mixture Model

CONTENTS

1 INTRODUCTION

1.1 Motivation

Every day we are expected to make choices. What cloth to wear, which book to read or which song to listen to. The choices available to choose from are frequently immense. Recommender systems can prove very helpful with these tasks. The most popular algorithm for recommending items has been collaborative filtering for many years (UITDENBOGERD; SCHYNDEL, 2002). It makes recommendations based on user data. Using user data to generate recommendations is an efficient and smart way to achieve many good recommendations but lesser or unknown items will never be recommended. In the last years content-based recommendation for music has grown more popular. Digital music archives have been growing ever since with new countless sub genres appearing each year. Many listeners know that there are many unknown but amazing artists hidden in the waste amount of music data available. The motivation of this work is to recommend music not based on their popularity amongst others, but based on their content and shaped to the user's taste, supporting lesser known producers and DJs as well as listeners who have a more unusual taste.

1.2 Goals

This work's goals are to prototype a recommender system based on audio analysis and evaluate it's performance. The system should be able to provide quality recommendation using only the liked tracks of a particular user and the available audio data. Furthermore the system is constructed to have 100% coverage over all tracks in its database. This work will examine if this is feasible and how useful it would be to implement such a system.

1.3 Structure

Five chapters divide this work. In the introduction, motivation and goals are explained. The second chapter examines traditional approaches, related fields and some background techniques. The third chapter explains the approach used in this work. It talks about how the data was obtained and which techniques will be used. The fourth chapter treats the evaluation of the system. Finally in the last chapter conclusions are presented and as well as thoughts about

future work.

1.4 Concepts, Vocabulary and Notation

Unless noted otherwise, with the term artist the performer of a musical piece is meant not the composer. With track we refer to a concrete recording of a musical piece.

Recommender systems usually operate on a information domain, which consists of users and their preferences for various items. A single preference is often expressed as a tuple (User, Item). Sometimes this preferences also include ratings which can be binary or integer scales but for reasons of simplicity this work will consider only unary preferences. If a certain user likes some item we will refer to this as a like. As we are working with a music recommender our items will be single music tracks or simply tracks.

Matrices and vectors are always bold letters, whereas scalar variables are normal letters. Indices are always displayed as minor normal printed letters.

As for some mathematical definitions we will introduce U as our set of users and T as a set of tracks. The like matrix $\mathbf{L}$ is composed by likes with $l_{u,t} = 1$ if the user liked the track and $l_{u,t} = 0$ if the user has not yet rated the track. $\bar{l}_u$ is the average of the user's u likes and respectively $\bar{l}_t$ is the average of track's t likes. Furthermore we define $\mathbf{\Pi}$ as the true and complete like matrix and $\mathbf{p}$ is the output we get from the recommender.

2 BACKGROUND

2.1 Overview

In this chapter we will review some background and related works. First the traditional collaborative filtering algorithm is explained. Subsequently some commonly known audio features are presented. Followed by some background on clustering techniques. Finally previous approaches in recommend music using neural networks are examined.

2.2 Collaborative Filtering

In the following two popular approaches for the CF algorithm are presented. Algorithms of this class usually need a big amount of user data to make good recommendations. This data is usually only available for popular tracks and therefore more popular tracks are recommended more frequently gaining more and more popularity. Tracks with not enough user data to be recommended appear stay unknown as they are rarely recommended.

2.2.1 User-based Collaborative Filtering

User-based collaborative Filtering were the first automated CF methods (EKSTRAND; RIEDL; KONSTAN, 2011). This approach tries to predict recommendation for a user A by inspecting similar user's likes and infer from their likes prediction for yet unrated tracks by A. User similarity is defined as having a similar likes. To compute such similarity a similarity function $s : U \times U \to \mathbb{R}$ has to be introduced.

2.2.1.1 Algorithm

To predict likes s is used to choose a set of nearest neighbours $N \subset U$. This set usually limited to 20 to 50 neighbours as this has been proofed to be a good number (HERLOCKER; KONSTAN; RIEDL, 2002). The next step is to infer likes from N. To do this we compute the average of the neighbours' likes weighted by their corresponding similarity. To compute like

predictions for a user u we get:

$$\boldsymbol{p}_{u,t} = \frac{\sum_{u' \in N} s(u,u') l_{u',t}}{\sum_{u' \in N} |s(u,u')|} \tag{2.1}$$

2.2.2 Item-based Collaborative Filtering

Item-based CF methods present a computational more efficient approach and are quite similar to User-based methods presented above. The difference is that instead of computing user similarity they compute track similarity based on user ratings. If two tracks are liked by the same users they are considered as similar. This clusters of similar tracks are then saved persistently to avoid repeating this expensive step for each recommendation. Furthermore big platforms tend to have more users than tracks. By having a sufficient high user to track ration a single user editing his likes is unlikely to have big impact on the similarity between items (EKSTRAND; RIEDL; KONSTAN, 2011). This allows to precompute similarity and allows the CF algorithm to scale to large user databases, while maintaining low computational costs. The similarity function need to be redefined to accept a tuple of tracks $s : T \times T \to \mathbb{R}$

2.2.2.1 Algorithm

After selecting a set of similar tracks $S \subset T$ similar to t. We compute the predictions as follows:

$$\boldsymbol{p}_{u,i} = \frac{\sum_{t \in S} s(t,t') l_{u,t'}}{\sum_{t \in S} |s(t,t')|} \tag{2.2}$$

2.2.3 Similarity Functions

As we use unary likes and define unrated tracks as 0, a good similarity function is the cosine similarity. It provides a simple, fast and good predictive function to estimate similarity between tracks as well as between users.

$$s(i,j) = \frac{\mathbf{L}_i \cdot \mathbf{L}_j}{\|\mathbf{L}_i\|_2 \|\mathbf{L}_j\|_2} \tag{2.3}$$

With $\mathbf{L}_i$ or $\mathbf{L}_j$ being either user likes or item likes according to which type of CF method is used. For different data domains there exists more appropriate similarity methods like e.g.

the Pearson Correlation (EKSTRAND; RIEDL; KONSTAN, 2011).

2.3 Audio Features

Substantial research has been carried out by the field of Music Information Retrieval a interdisciplinary field focused on retrieving information about music from digital environments, including audio data. We will present some of the most commonly used audio descriptors below but before we will present the short time fourier transform (STFT) as most features are based on it.

2.3.1 Short Time Fourier Transform

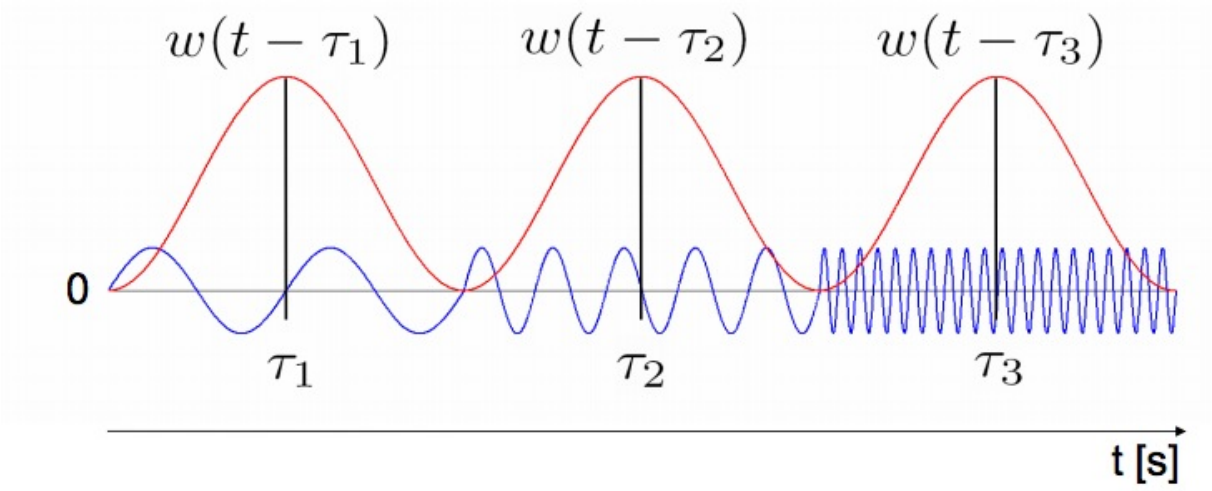
The Fourier Transform does not contain any information about how the spectrum of the signal changes over time. For music analysis the temporal aspect is quite important as music usually has a certain dynamic that has to be considered. For that reason the Short Time Fourier Transform is introduced. It's main difference to the Fourier Transform is that the signal $x(n)$ is splitted into multiple small segments τ_i, where the signal is considered as stationary (OHM; LÜKE, 2010). To improve results these segments may overlap. To remove artefacts we introduce a window function $\omega(n)$. A discrete fourier transform is applied to each of this segments. In the following the segments spectra will be referred to as frames. The STFT is then defined as follows:

$$\boldsymbol{X}(\tau_i, f) = \int_{-\infty}^{\infty} x(t)\omega(t - \tau_i)e^{-i2\pi ft}dt \tag{2.4}$$

Where τ is shifted by a fraction of the window function size depending on the amount of desired overlap.

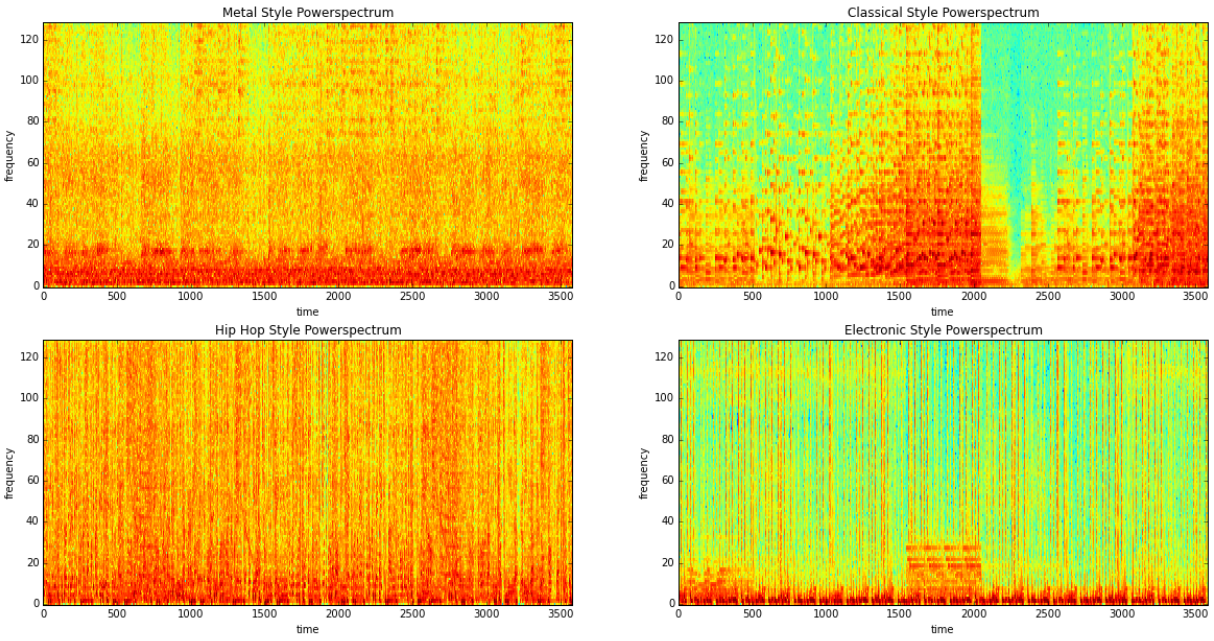
The magnitude of the STFT $|\boldsymbol{X}(\tau, f)|$is called spectrogram or power spectrum if the magnitudes have been squared and are often used to provide a graphical display of the spectrogram. Below we will see some examples of powerspectra. Notice the difference between different music styles. Metal and Hip Hop, styles with a big amount of vocals have high magnitudes spread over the whole spectrum and time while more instrumental based styles, like electronic or classical produce more unbalanced powerspectra with a tendency to higher magnitudes in the lower frequencies.

Figure 2.1: The figure below shows a signal (blue) and three window functions (red) with no overlap. For each segment the window function is multipied with the signal and a DTFT is applied.



Source: IFN Magdeburg

Figure 2.2: Figure showing logarithmic plots of powerspectra of different musical styles. The spectra were generated using a hamming window function, segment size of 23 milliseconds and an overlap of 50%.



2.3.2 Acoustical Features

2.3.2.1 Timbral Texture Features

Many subsequent feature use the magnitude of the power spectrum therefore we introduce $\boldsymbol{M}_t[n] = |\mathbf{X}(\tau, n)|^2$. Where $|\mathbf{X}(\tau, n)|^2$ is the squared magnitude at segment τ and frequency bin n. Subsequently some timbral texture features which are considered relevant by (TZANETAKIS; COOK, 2002) are presented. According to the author these features are standard features proposed for music-speech discrimination and speech-recognition.

2.3.2.1.1 Spectral Centroid The spectral centroid is a measurement for how bright or how dark a sound sounds. It is therefore a measurement of the spectral shape and is obtained by calculating the centroid of the magnitude spectrum of the STFT of a track.

$$C_\tau = \frac{\sum_{n=1}^{N} \boldsymbol{M}_t[n] * n}{\sum_{n=1}^{N} \boldsymbol{M}_t[n]} \tag{2.5}$$

2.3.2.1.2 Spectral Rolloff The Spectral Rolloff is a feature that measures how much energy resides in the lower frequencies. It is the frequency R_t below, which a certain fraction (commonly 0.85) of the magnitude distribution is concentrated.

$$\sum_{n=1}^{R_t} \boldsymbol{M}_t[n] = 0.85 * \sum_{n=1}^{N} \boldsymbol{M}_t[n] \tag{2.6}$$

2.3.2.1.3 Spectral Flux This feature is defined as the squared difference between the magnitude of two frames of the powerspectrum. The spectral flux is a measurement of local change in the powerspectrum:

$$F_t = \sum_{n=1}^{N} (\boldsymbol{M}_t[n] - \boldsymbol{M}_{t-1}[n])^2 \tag{2.7}$$

2.3.2.1.4 Time Domain Zero Crossings This feature describes the average of how many times the signal crosses the zero amplitude line. It is used as a measurement of noise in the signal.

The sign function returns 1 for positive numbers and 0 for negative numbers. $x(n)$ is the time-domain signal for segment τ

$$Z_\tau = \frac{1}{2}\sum_{n=1} N|sign(x(n)) - sign(x(n-1))| \tag{2.8}$$

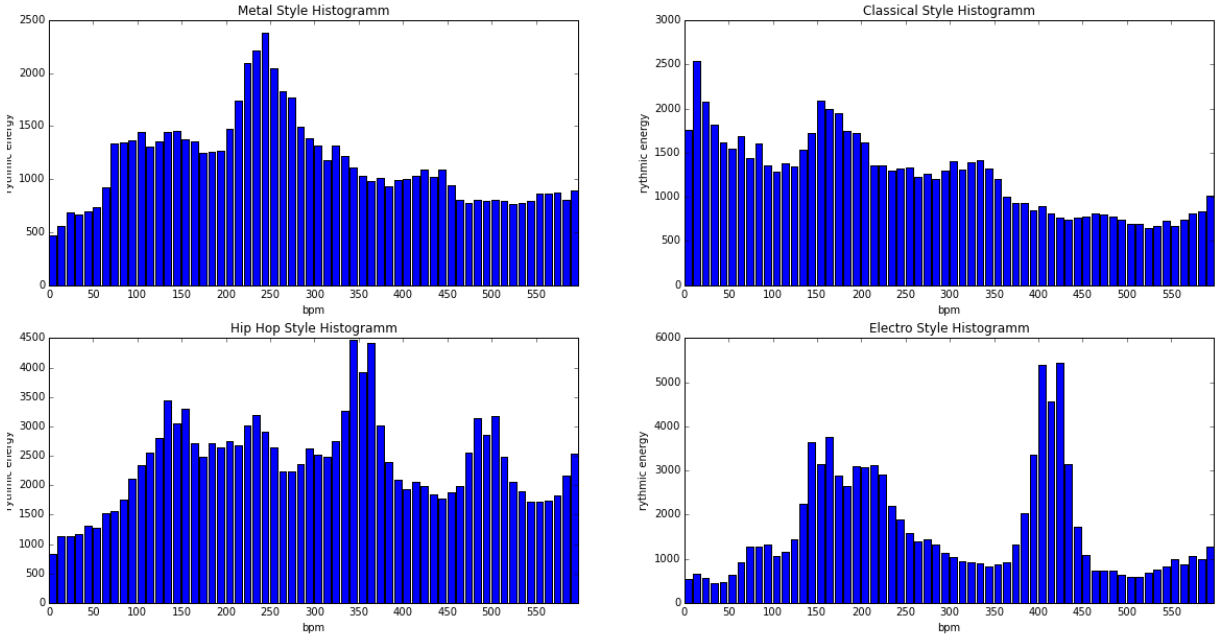
2.3.2.1.5 Mel-Frequency Cepstral Coefficients MFCC are features that are also based on the STFT. The FFT bins are transformed using triangular filter to fit into Mel Frequency bins. This features have been widely used in speech recognition systems but have also proofed very useful for musical timbre description(LOGAN et al., 2000). Later we will present how these features are motivated and calculated in detail.

2.3.2.2 Rhythm Features

2.3.2.2.1 Fluctuation Patterns Fluctuation Patterns also called Rhythm Patterns are psycho-acoustic, time independent features matrices which describe loudness modulation on various different frequency bands(RAUBER; PAMPALK; MERKL, 2002). These features have been adjusted to include research about the human perception of audio signals. How these are calculated and how research concerning human perception influences these features will be shown later in this work.

2.3.2.2.2 Rhythm Histograms Rhythm Histograms are based on Fluctuation Patterns and present a simpler representation of them(LIDY; RAUBER, 2005). In contrary to FP these feature do not store rhythmic information on different bands. The modulation magnitudes for all frequencies are summed up to obtain a histogram of rhythmic energy per modulation band. Figure 2.3 shows the rhythm histograms from the above shown STFT spectra. Observe as electronic style has two peaks at 150 and 400 bpm representing mostly bass and percussive features. Whereas the other features are more balanced, they still have their respective peaks at modulation bands typic for their genre.

Figure 2.3: Figure showing 4 example plots of rhythm histograms.



2.4 Clustering Techniques

Ensuing we will present some popular clustering techniques which are similar to the ones we will be using in our approach. Clustering means finding groups of data points in a multidimensional space, whose inter-point distances are small compared with distances to points of other groups(BISHOP, 2006).

2.4.1 K-means

K-means is a well known centroid based clustering technique. To explain this algorithm some notation is introduced. It will be assumed that the number of clusters K is given. For each cluster we define a set of vectors $\boldsymbol{\mu}_k$ with $k = 1, \ldots, K$. Let‘s assume now that these $\boldsymbol{\mu}_k$ represent the centres of our K clusters. The goal is to find an assignment of data points to clusters as well as find the best possible centres for these clusters. To describe the assignment of data points to cluster we introduce for each data point $\mathbf{x}_n$ a set of binary variables $r_{n,k} \in \{0, 1\}$, with $k = 1, \ldots, K$. If x_n is assigned to cluster k then $r_{n,k} = 1$ and $r_{n,j} = 0$ for all $j \neq k$. We

can than formulate an objective function:

$$J = \sum_{n=1}^{K} \sum_{k=1}^{K} r_{n,k} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|_2^2 \tag{2.9}$$

These represents the squared distance for each point to it's assigned cluster centre. To minimise the objective function two steps are made. By repeating this two steps convergence to a local minimum is assured(BISHOP, 2006):

1. Optimise the assignment of the data points to the centres by choosing the nearest center.

$$r_{n,k} = \begin{cases} 1 & \text{if} \quad k = \arg\min_j \left\| \mathbf{x}_n - \boldsymbol{\mu}_j \right\|_2^2 \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

2. Optimise position of centres. J is a quadratic function of μ_k to minimise this function we set it's derivative to zero and solve for μ_k so we obtain:

$$\mu_k = \frac{\sum_n r_{n,k} \mathbf{x}_n}{\sum_n r_{n,k}} \tag{2.11}$$

2.4.2 Multivariate Gaussian Mixtures Models

A mixture of gaussians is a probabilistic model that uses a superposition of K Gaussian densities. This distribution can be used to model almost any continuous density very accurately by using the right amount of components:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} | \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k) \tag{2.12}$$

as can be seen above each component has it's own mean $\boldsymbol{\mu}_k$ and it's own covariance $\boldsymbol{\Sigma}_k$. π_k are called the mixing components and govern the form of the distribution. As all probability density function have to be positive we can infer $\pi_k >= 0$ and by integrating 2.12 we get:

$$\begin{aligned} \int p(\boldsymbol{x}) d\boldsymbol{x} &= \sum_{k=1}^{K} \pi_k \int \mathcal{N}(\boldsymbol{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) d\boldsymbol{x} \\ 1 &= \sum_{k=1}^{K} \pi_k \end{aligned} \tag{2.13}$$

as the integrals of probability densities are normed to be 1, we get a further constraint. It

holds that $0 \leq \pi_k \leq 1$ hence π_k can be considered a probability itself and is known as the prior probability of picking the k^{th} component.

Now to find the right parameters of such a mixture distribution to fit the data we can maximise the logarithmic likelihood over all N data points $\boldsymbol{X} \in \mathbb{R}^N \times \mathbb{R}^D$, which is given as:

$$\ln p(\boldsymbol{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} ln(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)) \tag{2.14}$$

This formula is kown to have no closed-form analytical solution (BISHOP, 2006) and can therefore only be estimated using an algorithm known as expectation maximisation. In this work we will use a neural network called Incremental Gaussian Mixture Network to estimate the number of components and their respective parameters for each component automatically. It also uses the expectation maximisation algorithm and some heuristics to add or remove components incrementally. For more details and explanations on this algorithm (HEINEN; ENGEL, 2010) should be consulted.

To use a Gaussian Mixture Model for clustering it is sufficient to assign each datapoint, after fitting the parameters, to the component corresponding to the highest posterior probability $p(k|\boldsymbol{x})$, as given by the Bayes' theorem:

$$p(k|\boldsymbol{x}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)}{\sum_l \pi_l \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_l, \boldsymbol{\sigma}_l)} \tag{2.15}$$

2.5 Psychological and Human Aspects

2.5.1 Music Similarity

As seen above to give recommendation, a definition of similarity is inevitable. Many studies have been made concerning music similarity. It was soon noticed that human perception of music similarity does not comply with the definition of an Euclidean metric(BERENZWEIG et al., 2004). Furthermore music similarity is perceived very subjective by most individuals. Often the perception of similarity is influenced by current mood as well as by the amount of interested someone has in a certain style of music. It is often perceived that people who aren't interested in a certain style perceive all music as "the same". Moreover music has many distinct dimensionalitys such as tempo, rhythm, tonality, genre, melody, geographical origin, lyric content, and much more (BERENZWEIG et al., 2004). This permits music similarity to be interpreted on many different dimensions, adding complexity to the problem. In this approach

we will limit these dimension to acoustic similarity.

2.5.2 Personality and Cultural Background concerning Musical Perception

Most of us have experienced how cultural background or personality influence musical taste and perception. For example there has always been a cultural bias against popular music. Nowadays lesser than 50 years ago. Some research concerning preferences in musical styles, published before 1950, contains statements that defines popular music as "music that is ranked by critics as tawdry, banal, insipid"(GERNET, 1940).

Also personality influences preferences in musical styles. In 1939, Burt published research which subjected individuals to introvert versus extrovert and stable versus unstable personality tests as devised by Eysenck. He came to the conclusion that stable extroverts prefer solid predictable music in contrast to stable introverts who enjoy listening to more cognitive classical and baroque styles. It is also interesting that a study conducted in Japan shows that japanese adolescents are more likely to enjoy listening to classical or jazz music than Americans of the same age (WELLS; TOKINOYA, 1998). It was also shown that heavy metal music is generally disliked because of social stigmata associated with it. Another study shows that there are tempo preferences between social groups (UITDENBOGERD; SCHYNDEL, 2002). It was also shown that women are more interested in music than men. Moreover it has been shown that people prefer musical styles which they were listening to during a critical period of their lives at an average age of about 23.5 years. Hence it can be seen that there exist a lot of factors which might influence how music is perceived differing form individual to individual.

2.6 Related Approaches

It should be mentioned that there exists other techniques besides analysing audio data. Noteworthy is the work (VEMBU; BAUMANN, 2005) which tries to make music recommendation by textual mining album reviews. Another interesting approach is to use social tagging websites to retrieve tags for tracks, like done by (HARIRI; MOBASHER; BURKE, 2012). Using data generated by humans about a musical piece allows to include personal and cultural context into features, which can be very useful as these backgrounds affect how music is perceived. (YOSHII et al., 2008) could achieve very good results combining CF models with content based approaches. This work will focus on a content based approach and evaluate more

complex feature descriptors extracted form music. Furthermore the evaluation will be done on a bigger and less biased dataset across all genres.

3 THE APPROACH

In this work it is assumed that acoustic similarity can be used to recommend musical pieces to a user. It is also assumed that the music similarity can be modelled using an euclidean metric for simplicity reasons. The approach presented here makes use of audio features as used in MIR which have been shown as being good descriptors of musical features. These features are then compressed in their dimensionality and presented to a SOM. After training the network we can assign each track in our database to a cluster. New unknown tracks can also be clustered immediately as the neural network has already been trained using our current database. Hence allowing this recommender system to work online. New tracks can therefore be recommended instantly without having to wait for sufficient user data about them being available. This clustering is then used as a similarity measure between tracks. We then examine a users previous likes and their relative distribution among these clusters to give recommendation on similar tracks. A brief overview over the whole system is shown in Figure 3.12

3.1 Data Acquisition

It proofed quite difficult to obtain a dataset for this approach. Most datasets provided for music information retrieval are very small and provide no user data at all. Furthermore they are often limit to a unrealistic number of genres. We will describe how a realistic dataset was constructed to provide audio as well as user preference data.

3.1.1 Data Sources

To obtain audio data as well as user data for evaluation, a subset of the Million Song Database (BERTIN-MAHIEUX et al., 2011) was used as it provides sufficient metadata as well as track ids from services like 7digital to obtain preview audio. Furthermore we used the ThisIsMyJam archive dump(JANSSON; RAFFEL; WEYDE,) to get user data on tracks. ThisIsMyJam was a social music website where users could post their current favourite track from various sources. It was possible to follow other users as well as to like other users jams. We preferred this dataset over the TasteProfile dataset. As it contains explicit likes from users in contrary to listening history which also presents preferences but in a implicit form.

3.1.2 Constructing the subset

As Audio as well as user data is needed we constructed a subset of the MSD to include all these requisites. This subset was constructed as follows:

1. The ThisIsMyJam dump likes file was constrained to obtain a list of likes which refer to songs known to the MSD. To achieve this we used a mapping from jam ids to MSD ids provided by the LabROSE.
2. To limit computational costs we randomly sampled 250 users from this data. This 250 users made 14489 likes which matched to 8496 distinct MSD tracks.
3. To get audio data for each track the 7digital API was queried by preferring track ids obtained from the MSD if available. If not available album id, artists ids or text search where used to find the correct audio file. 7digital provides preview audio files with a length of 30 to 60 seconds. After querying it was possible to obtain 8003 audio files.
4. we restrict our subset of likes and tracks to the tracks, for which audio previews could be obtained.

Finally the subset consists of 250 distinct users with 13891 likes and 8003 distinct tracks with audio files.

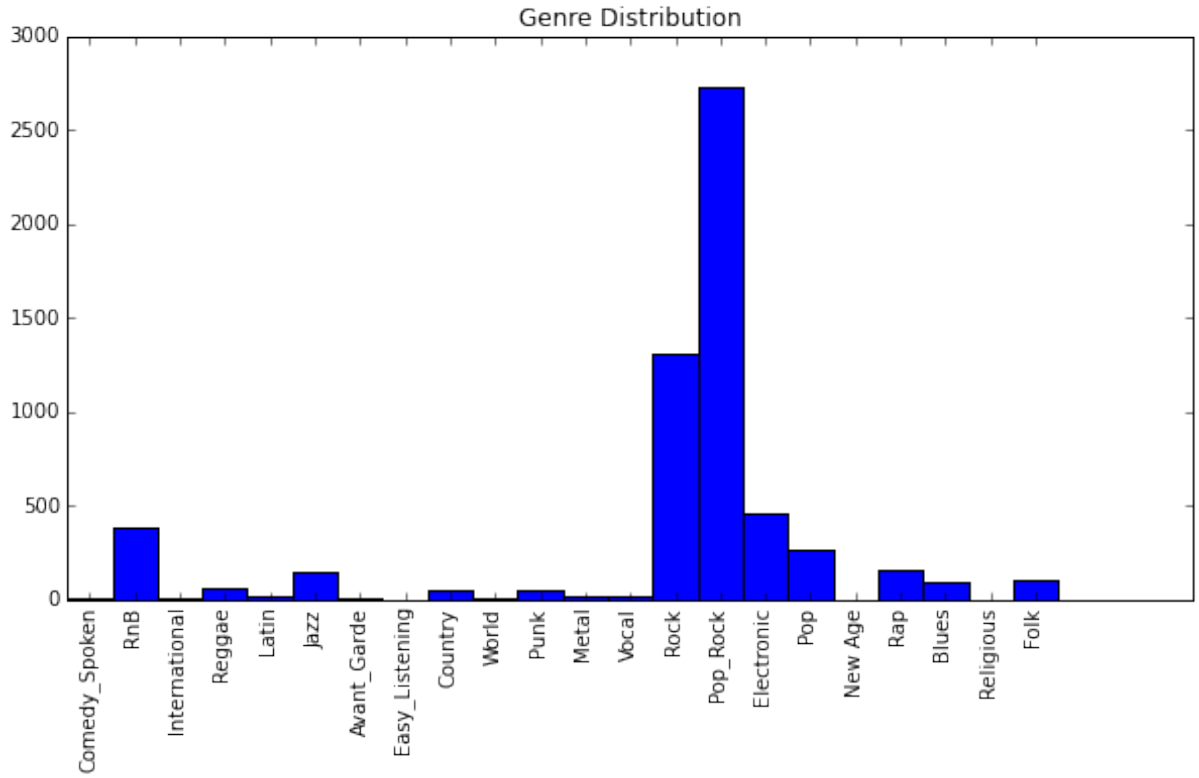
3.1.3 Genre Distribution

For about 6000 tracks genre attributes could be obtained using the tagtraum genre annotations (SCHREIBER, 2015) as well as the MAGD dataset(TU-WIEN, 2014). In Figure 3.1 we can see that the data is very unbalanced and almost 50% are Pop/Rock songs.

3.2 Audio Features

The features chosen to form feature vectors for our recommender system are Fluctuation Patterns and Mel-Frequency Cepstral Coefficients as they are popular in other research fields and have been proven to provide good results for most musical domains.

Figure 3.1: Genre annotations for 6000 tracks of the constructed subset



3.2.1 Preprocessing Audio

To reduce computational complexity some preprocessing is done to reduce the input data. First the raw audio data is combined into one mono channel. Then the audio is sampled down to $11025Hz$. Next the data is segmented into $6s$ chunks and intro and outro chunks are removed. If the audio track is longer than $30s$ every third chunk is removed.

3.2.2 Mel-Frequency Cepstral Coefficients

MFCC is short for Mel-Frequency Cepstral Coefficients. These are short-term spectral based features. This means they usually are time dependent and offer therefore a description of timbre evolution over time. Although MFCCs were designed for speech recognition they also proofed to be very useful in music analysis (LOGAN et al., 2000). The Computation of MFCCs is oriented at the model of speech production which claims that speech is modelled by a linear system. The excitation signal, generated by the vocal cords, is thereby enveloped by the impulse response of the vocal tract. For speech-recognition the excitation signal is irrelevant. Important

is only the slow change of the vocal tract which determines the spoken sound. The folding is converted into an addition by taking the cepstrum and can therefore be separated more easily from the excitation (NIEMANN, 2013).

1. **Calculating the Powerspectrum**

   The powerspectrum of the preprocessed audio is computed by squaring the spectrum obtained from STFT as described in 2.3.1, using a segment size of 256 samples which corresponds to 23ms and a Hanning window with 50% overlap, yielding the matrix $\boldsymbol{X}$.

2. **Taking the logarithm**

   Little is found about the motivation of this step in literature. Most probably it accounts for the phenomena that loudness is not perceived linear.

$$\boldsymbol{X}'(i,j) = \log(\boldsymbol{X}(i,j)) \tag{3.1}$$

3. **Map to Mel Scale**

   The coefficient of the logarithmic power spectrum X' are summated into mel bin using triangular filter banks. The correlation between frequency and perceived pitch is none linear and can be expressed by:

$$f_{mel}(f_{Hz}) = 1125 * \log(1 + \frac{f_{Hz}}{700}) \tag{3.2}$$

   To obtain the triangular filter bank a linear spacing defining the desired number of mel bins $N = 40$ is done in the mel domain resulting in a vector $\boldsymbol{l}_{mel}$ containing upper and lower limits of the bins. This vector is then transformed back into frequency domain using the inverse of the above function f_{mel}. To compute the right indexes a helper function $binfreq : [1, \ldots, 256] \to \mathbb{R}$ has to be introduced. This function shall return the associated frequency considered with the bin index.

$$\begin{aligned} \boldsymbol{l}_{Hz}(i) &= f_{mel}^{-1}(\boldsymbol{l}_{mel}(i)) \\ idx_i &= \{j | binfreq(j) \geq \boldsymbol{l}_{Hz}(i) \wedge binfreq(j) < \boldsymbol{l}_{Hz}(i+1)\} \\ \boldsymbol{C}_{mel}(i,j) &= \sum_{k \in idx_i} \boldsymbol{X}'(k,j) \end{aligned} \tag{3.3}$$

4. **Apply the DCT**

   The frames in $\boldsymbol{C}_{mel}$ are usually highly correlated. To reduce the number of parameters usually a Discrete Cosine Transform (DCT) is applied to each frame to decorrelate the

components. Theoretically the DCT is an approximation to the Principal Component Analysis and is used as it is fast and domain independent. Only the first 13 cepstral feature are kept from each vector.

3.2.3 Fluctuation Patterns

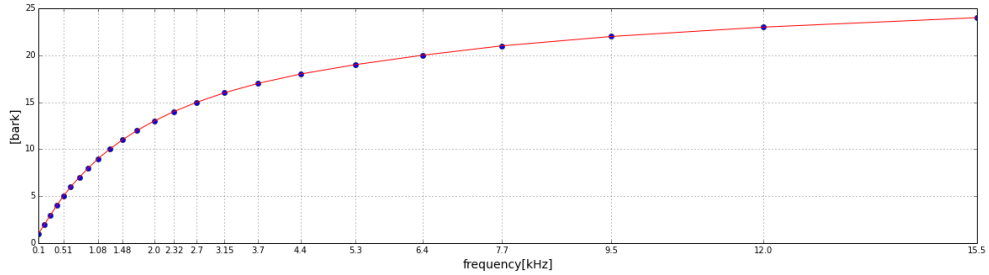
As mentioned above Fluctuation Patterns provide a time independent feature for music rhythm description, while considering characteristics of human auditorial perception. It follows a description of multiple steps on how to compute these features and how these steps are motivated. The process of computing fluctuation patterns can be split into two major parts: Calculating the Specific Loudness Sensation and computing the Rhythm Patterns, as proposed by (PAMPALK, 2001).

3.2.3.1 Specific Loudness Sensation

In this step some psychoacoustics scales and filter are applied to the STFT of the preprocessed audio data. This involves 6 steps:

1. **Calculating the Powerspectrum**
   The powerspectrum of the preprocessed audio is computed by squaring the spectrum obtained from STFT as described in 2.3.1. Using a segment size of 256 samples which corresponds to 23ms and a Hanning window with 50% overlap.
2. **Bark Critical Bands**
   According to (FASTL; ZWICKER, 2007) our inner ear separates certain frequencies and concentrates them on certain parts of our basilar membrane. So the inner ear can be regarded as a set of asymmetrical bandpass filters. These bandpass filters have centre frequencies related to the ones of the bark scale. The position and width of these bands has been analysed in psycho-acoustical experiments by using a loud tone to mask a quiet one. At high frequencies the frequency difference of these two tones must be greater so that listeners can hear the quiet tone. At lower frequencies the quiet tone can be perceived at smaller differences. Moreover these masking effects are closely related to noticeable frequency variations. Therefore two tones within a critical band are very hard to distinguish. Listeners usually can't tell which one of the two is higher or lower.

Figure 3.2: Bark scale



3. **Spectral Masking**

As mentioned above a louder tone can mask a quieter one, if they are played simultaneously or shortly before or after. This can also occur across critical bands and is a natural phenomena that helps to reduce noise. A spreading function has been proposed to account for the effects of simultaneous masking(SCHROEDER; ATAL; HALL, 1979; PAMPALK, 2001). The spreading function is applied to each frame of our critical band rate spectrum matrix and describes the influence of the i^{th} band on the j^{th}, visualised in Figure 3.3:

$$\mathbf{S}(i,j) = 15.81 + 7.5(i-j+0.474) - 17.5\sqrt{1+(i-j+0.474)^2} \tag{3.4}$$

To apply the spreading function to each frame of our matrix we can simply multiply the spreading matrix with the critical band rate spectrum matrix $\mathbf{B}$:

$$\begin{aligned} \mathbf{B}_s(i,t) &= \sum_{j=1}^{24} \mathbf{S}(i,j)\mathbf{B}(j,t), \text{this is equivalent to} \\ \mathbf{B}_s &= \mathbf{SB} \end{aligned} \tag{3.5}$$

In Figure 3.4 is demonstrated how spectral masking affects the critical bands. Higher frequencies are affected more than lower ones, these can be observed especial in the profile plot. The two right plots of the vertical matrix demonstrate that it only affects single frames. This step has been shown to controversial as it might have negative influence on the performance of this feature (LIDY; RAUBER, 2005).

Figure 3.3: Logarithmic plot of S, notice the asymmetrical shape

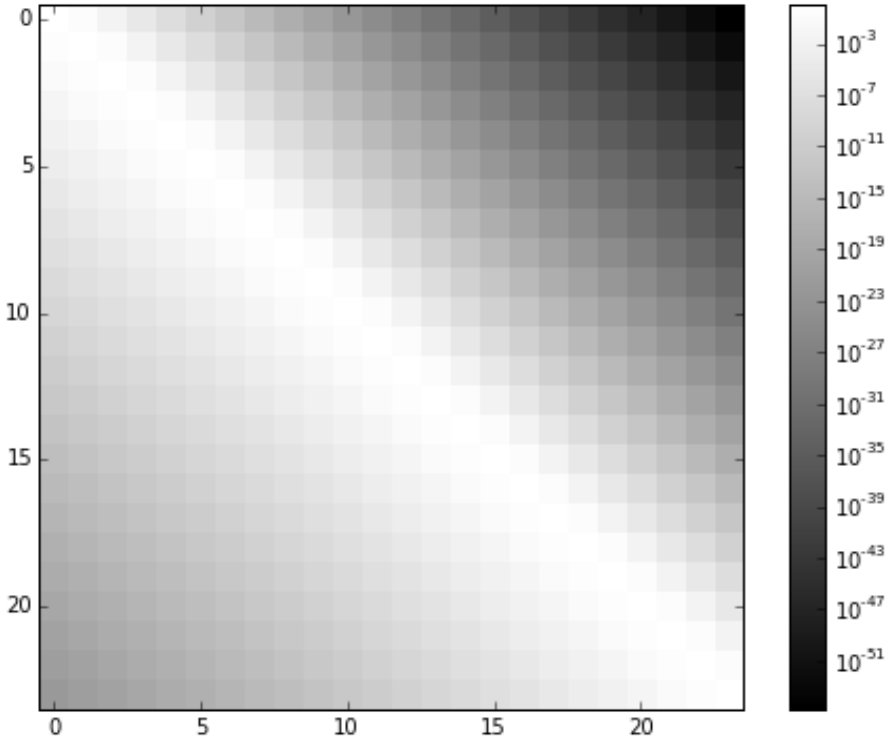
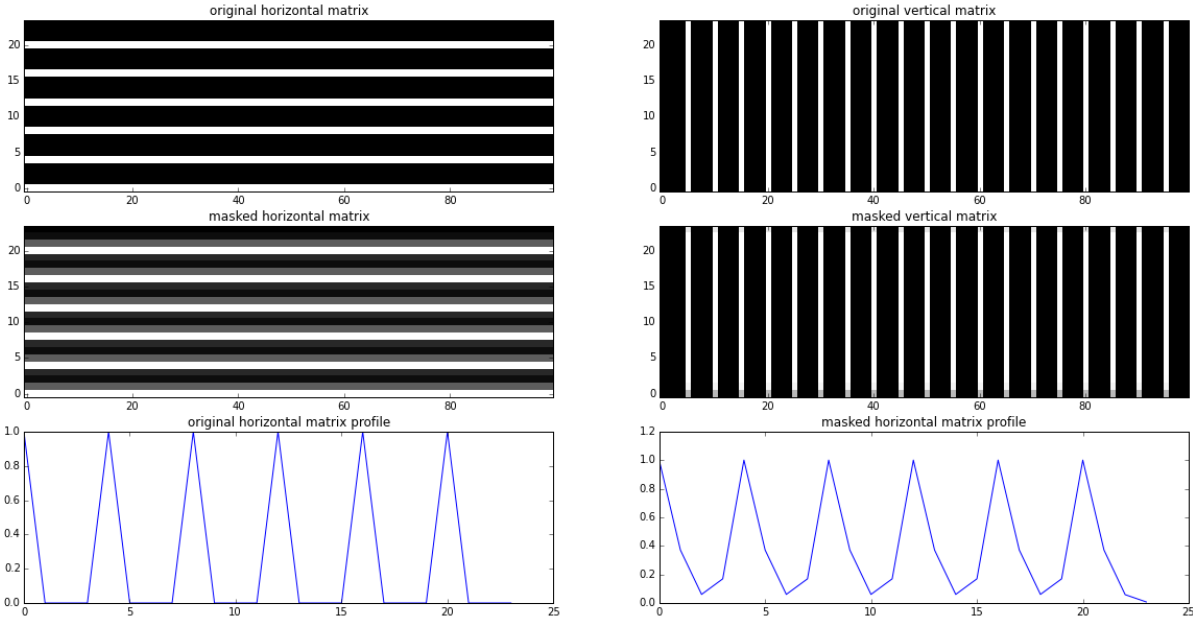


Figure 3.4: Examples of how spreading affects the critical bands



4. **Decibel Scale**

   To proceed the values of our matrix B_S have to be converted into the decibel scale. The decibel is a logarithmic scale which measures the ratio between two numbers. As the human hearing is able to perceive a rather large range from the lowest to the loudest sound, in terms of sound pressure level, approximately $10^{12} : 1$. The decibel scale offers a much more convenient way to represent these numbers by mapping those to a more manageable range between 1 and 140. As the decibel scale is a relative scale it uses instead of a zero point, a reference point, which can be arbitrary. Sound amplitudes are most often measured using sound pressure levels (SPL) as are our digital audio archives based on SPL which measures amplitudes in terms of $dynes/cm^2$ or Pascal. Usually the reference point used for this unit is $20\mu Pa$ as these represents the hearing threshold of a sound at $1000kHz$ (LASS, 2012) which is, as already mentioned, quite popular in acoustics. To calculate decibels based on sound pressure levels from audio archives some adjustments have to be made. As our data is represented in bits we will define the lowest possible sound as 1 (or -1, as we use magnitudes). The conversion from SPL to decibel is given by:

$$dB(SPL) = 20 * \log(\frac{p}{p_0}) \tag{3.6}$$

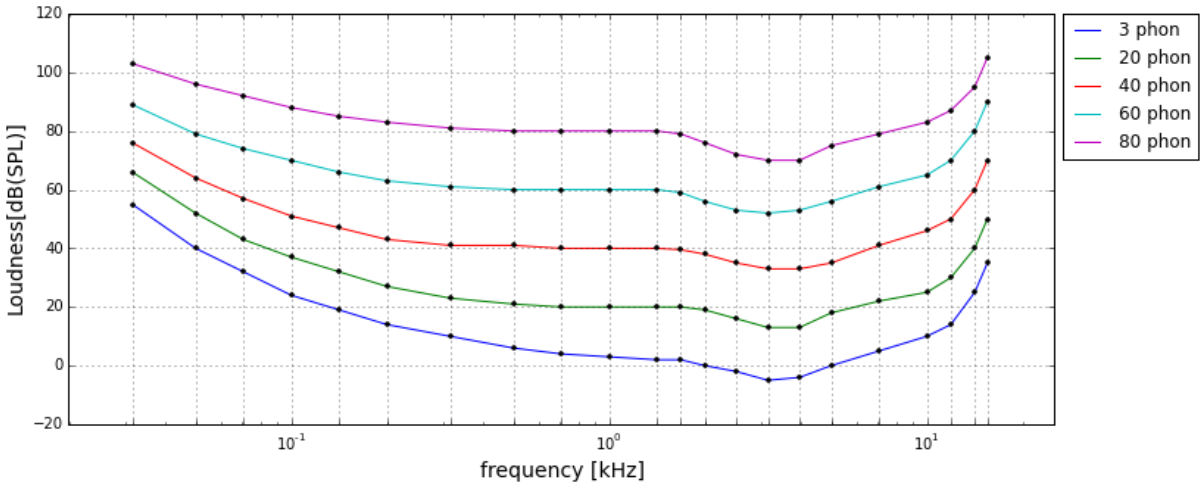
   Whereas p_0 is the hearing threshold defined as 1. By applying this formula to each entry of our matrix $\mathbf{B}_S$ a loudness matrix is obtained. It is necessary to define all entries lower than one to 1 to avoid the singularity of the $\log$ function.

$$\mathbf{B}'_S(i,j) = \begin{cases} 1, & \text{if} \quad \mathbf{B}_S(i,j) < 1 \\ \mathbf{B}_S(i,j) & \text{otherwise} \end{cases} \tag{3.7}$$
$$\mathbf{L}_{dB}(i,j) = 20 * \log_{10}(\mathbf{B}'_S(i,j))$$

5. **Phon Scale**

   How we perceive loudness at different frequencies is not linear. To account for this effect the phone scale is introduced which allows to map our matrix's values to phon. The phon makes loudness sensation independent from the frequency. A tone at any frequency with 40 phon is defined to be as loud as a 40dB tone at 1kHz. The equal loudness contours in 3.5 have been obtained by experiments involving single tones. These contours achieve a minimum around 2kHz to 5kHz which are the frequencies humans are most sensible to. Outside these interval the curves rise rapidly corresponding to decrease in hearing

Figure 3.5: The phone contours



sensibility outside this interval. To obtain the phon representation $\mathbf{L}_{phon}$ of the loudness matrix $\mathbf{L}_{dB}$, we define the equal loudness contour matrix $\mathbf{C}_{elc}(i,j)$ which contains the in Figure 3.5 plotted curves, with the decibel value of j^{th} contour at the i^{th} critical band (represented as black dots in Figure 3.5). The corresponding phon values can then be interpolated, using $\mathbf{c}_{phon} = [3, 20, 40, 60, 80, 100]$ as follows:

1. Cast values below contour of phon 3 to smallest available value.

$$\mathbf{L}'_{db}(i,t) = max(\mathbf{L}_{db}(i,t), \mathbf{C}_{elc}(i,1))$$

2. for each entry find upper contour index

$$level_{i,t} = \arg\min_{j}(\mathbf{L}'_{db}(i,t) < \mathbf{C}_{elc}(i,j))$$

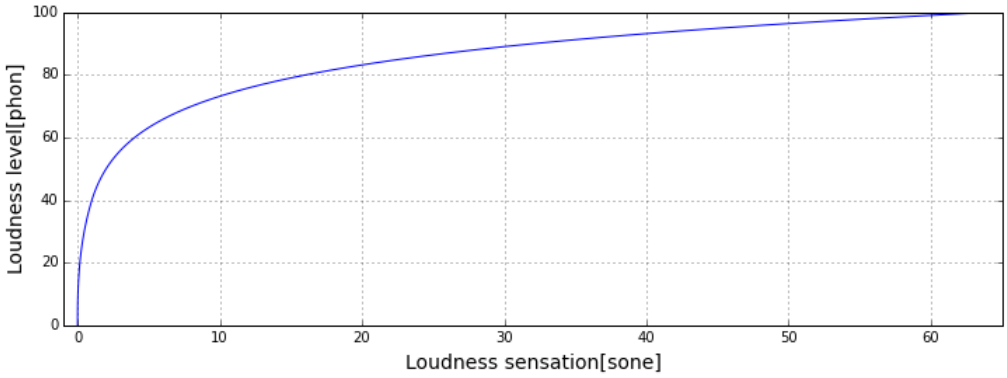
3. Linear interpolation first step: $r_{i,t}$ being the relative distance to the lower level

$$r_{i,t} = \frac{\mathbf{L}'_{db}(i,t) - \mathbf{C}_{elc}(i, level_{i,t} - 1)}{\mathbf{C}_{elc}(i, level_{i,t}) - \mathbf{C}_{elc}(i, levels_{i,t} - 1)}$$

4. Linear Interpolation second step: compute interpolated value

$$\mathbf{L}_{phon}(i,t) = \mathbf{c}_{phon}(level_{i,t} - 1) + r_{i,t}\mathbf{c}_{phon}(level_{i,t})$$

Figure 3.6: Relation between loudness sensation and loudness level



5. **Sone Scale**

   Notice that phon scale is not linear as doubling the phon does not result in a doubled loudness perception. To transform the phon unit into a linear scale the sone scale is used, which represents perceived loudness in a linear scale. How the phone scale relates to the sone scale can be seen in Figure 3.6. The conversion is calculated as follows:

$$\boldsymbol{L}_{sone}(i,t) = \begin{cases} 2^{\frac{1}{10}(\boldsymbol{L}_{phon}(i,t)-40)} & \text{if} \quad \boldsymbol{L}_{phon}(i,j) > 40 \\ (\frac{1}{40}\boldsymbol{L}_{phon}(i,t))^{2.642} & \text{otherwise} \end{cases} \tag{3.8}$$

3.2.3.2 Rhythm Patterns

Until now the obtained feature matrix from last section $\boldsymbol{L}_{sone}$ is still time dependent. Shifts in the signal of the same music would result in highly dissimilar matrices. Making this feature vector unsuitable as a descriptor. Hence Rhythm Patterns are introduced to obtain a time invariant representation of the loudness sensation.

If we examine the loudness of a specific band we will observe that it rises and falls several times. This is likely to occur in a periodical pattern, which is perceived as the rhythm. Usually instruments follow a rhythmic pattern, which is very accurately timed, as perceivable in the critical bands. This can be seen in Figure 3.7m which depicts a plot of several critical bands using $\boldsymbol{L}_{sone}$ of an electronic music track. As we can see there is some periodicity especially in the lower bands as electronic music uses mostly simple rhythms at low frequencies (commonly known as beats). It is also noteworthy to examine all these critical bands overlaid in a single plot, as peaks repeatedly meet at a certain periodicity, which can be interpreted as

a musical bar, a time segmentation most musicians use to time their compositions. If we consider this periodical pattern as a linear combination of sinuids in a discrete sampled domain we can use the FFT to reconstruct the amplitude corresponding to each frequency for each individual critical band. A 3-step approach, proposed by (RAUBER; PAMPALK; MERKL, 2002), is presented subsequently to construct time independent rhythm descriptors with psycho-acoustic considerations.

1. **Amplitude Modulated Loudness**

   As described above each critical band is likely to have some periodic phenomena. Consequently we will analyse the underlying modulation frequencies. These analysis is done for each 6 second segment, described in 3.2.1. As a result of the STFT we have time quanta of $12ms$ (considering the overlay of the windows). Applying a FFT on each of this 6 second segments will result in a spectrum with frequency bin from $0Hz$ to $43Hz$ with an accuracy of $0.17Hz$. Note that a modulation frequency of $43Hz$ corresponds to approximately 2600 beats per minute (bpm). The modulation amplitude is calculated as follows:

   $$\boldsymbol{\Delta L}_s(i) = FFT(\boldsymbol{L}_{sone}(i, s * [1, \ldots, 512])) \tag{3.9}$$

   With $\boldsymbol{L}_{sone}(i, s * [1, \ldots, 512])$ being the i^{th} critical band of s^{th} $6s$ segment.

2. **Fluctuation Strength**

   the hearing sensation produced by slow amplitude fluctuation, usually less then $20/s$, is called fluctuation strength. Sensation induced by faster fluctuations are called roughness (FASTL, 1982). It was discovered that fluctuation strength is most intense around modulation frequencies of $4Hz$ an gradually decreases until reaching $15Hz$. After $15Hz$ the sensation of roughness starts to increase. It was shown that the absence of roughness is a prerequisite for musical consonance (FASTL, 1982). To take the unequal sensation of fluctuation strength into account a weighting function $f : \mathbb{R} \rightarrow [0; 1]$ based on Fastl's model is introduced an shown in Figure 3.8. With this function fluctuation modulations around $4Hz$ are emphasised.

   $$\boldsymbol{F}_s(i, j) = \boldsymbol{\Delta L}_s(i, j) * f(j) \tag{3.10}$$

Figure 3.7: Loudness modulation in certain critical bands of an electronic music piece. Observe the periodicity especially in lower bands and in the last plot.

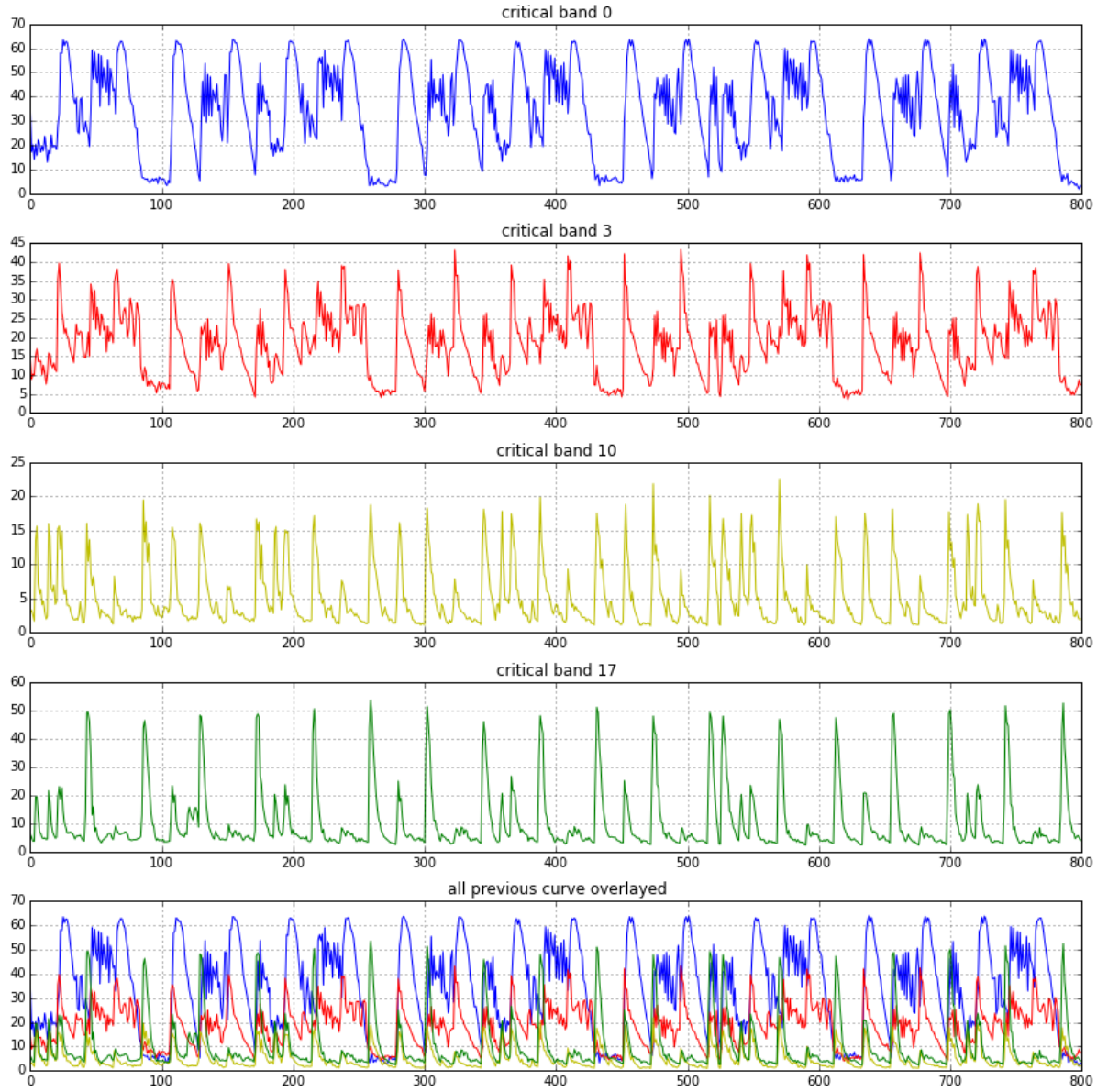
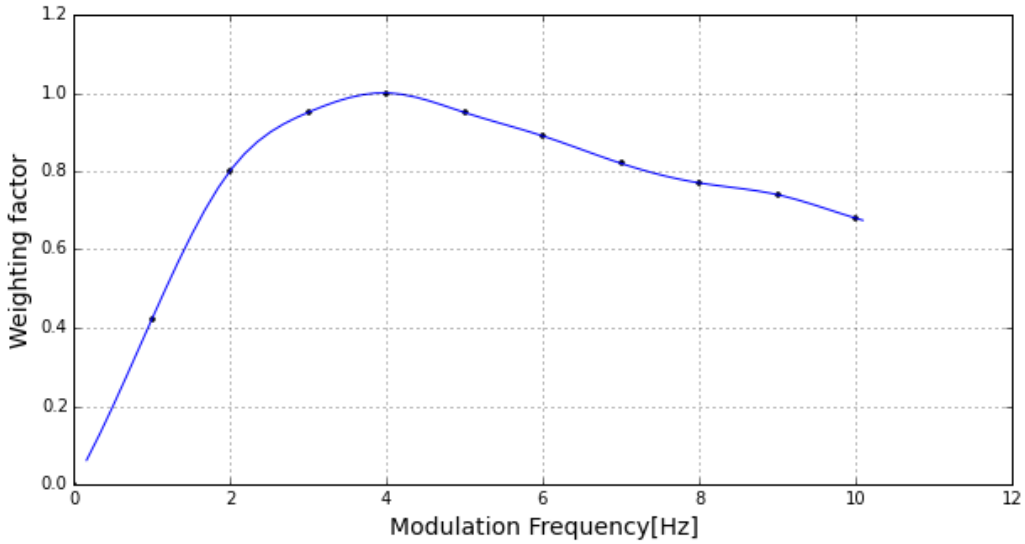


Figure 3.8: fluctuation strength model



As after $15Hz$ roughness starts to increase by plotting a summary of all data of fluctuation patterns, it can be observed that after $10Hz$ there is hardly any activity (PAMPALK, 2001). Consequently only the first 60 values which correspond to $10Hz$ are used.

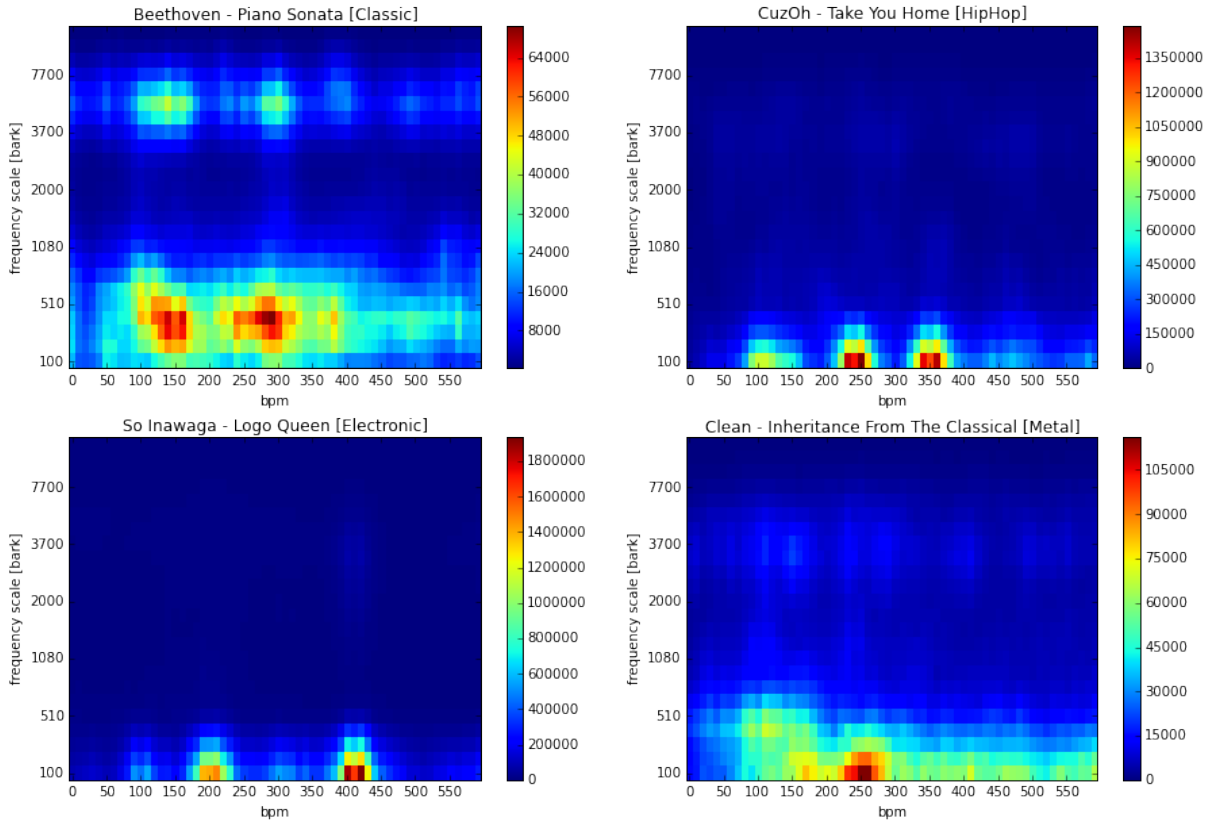
3. **Modified Fluctuation Strength**

   At this point we have time independent rhythm descriptors which would already provide good results. To further improve this features (RAUBER; PAMPALK; MERKL, 2002) applies gradient filter to enhance vertical lines in the matrices which occur in beat rich music styles. Afterwards a gaussian filter is applied to blur across critical bands as the exact bands are irrelevant for music similarity. This measures reduce the Euclidean distance between two similar music pieces, which is a desired effect.

3.2.3.3 Examples of Modified Fluctuation Patterns

In this section we will present some examples of fluctuation patterns and discuss their similarities and distances in music space. Figure 3.9 shows 4 fluctuation pattern matrices of 4 different tracks each of a distinct musical style and Table 3.1 shows the distances between the corresponding normalized and compressed fluctuation patterns. Observe how Hip Hop and Electro tracks produce kind of similar fluctuation patterns as they both have a very bass and beat based rhythms. They still are distinct as the Hip Hop track has a lot of fluctuation also in higher bands but they are compared to the beat loudness rather small and spread over all frequencies therefor less visible in the plot. The Electro track has only a smaller peak at around

Figure 3.9: Examples of Fluctuation Patterns



$400bpm$ in higher frequencies, which are percussive instruments like hi-hats. Therefore they are still distinct enough as can be seen in distance matrix they have the biggest distance between all pairs. It might seem surprisingly that metal and classic music have the lowest distance between each other, but observing the fluctuation patterns we can see that they are quite similar in rhythmic composition. Beethoven's Piano Sonata is a quite lively piano pieces with many different notes played in fast sequences with varying loudnesses. This can be interpreted as similar to a Metal track which usually has a lot of guitar chords played in fast sequence. Of course the metal track's modulation frequencies reside in lower frequencies which is clearly visible in the plot. Still there are quite some differences for example the classic tracks has many peaks whereas the metal track as a single beat around $250bpm$ which is most probably the bass drum. Also the scales are very different the metal track is perceived almost double as loud as can be seen on the values on it's colour scale.

Table 3.1: Distance matrix between sample tracks

	Classic	Electronic	Metal	HipHop
Classic	0.000	5.824	1.694	4.584
Electronic	5.824	0.000	5.782	6.215
Metal	1.694	5.782	0.000	4.608
HipHop	4.584	6.215	4.608	0.000

3.3 Feature Combination

Now there exists a Fluctuation Pattern matrix for every 6s chunk of every track in the database and for every 6 second chunk exists also a matrix with MFCC containing a timbre vector for every $23ms$. As these data amounts are still to big to handle for most clustering algorithms the data has to reduced. It arises the question: from the 6s chunks which one represents best the songs rhythm and how can we express the song's timbre dynamically without needing big amounts of data? In the following section we will present various approaches trying to achieve this. In the Evaluation chapter we will see how these different strategies preform.

3.3.1 Fluctuation Pattern Dimensionality Reduction

The Fluctuation pattern matrix has dimenionality of 1440 as we have 24 frequencies in the first dimension and 60 modulation frequencies in the second dimension. As this is to big to achieve a stable clustering, the data is reduced using PCA to a dimensionality of 60.

3.3.1.1 Chunk Selection

As shown by (PAMPALK, 2001) the simplest and quite good preforming method to compress all chunks is to calculate the median between all $6s$ chunks. Leaving a single fluctuation pattern matrix to be further compressed using PCA.

3.3.1.2 Principal Component Analysis

The principal component Analysis or PCA is a method used here for dimensionality reduction. The PCA is a projection of data of dimensionality D into a subspace of dimensionality M while minimising the average projection cost which is defined as the squared distance between data points and their projections. To calculate the PCA we have to select a basis of vectors

so that the cost function is minimised. To do this let's assume we have complete orthonormal basis of D-dimensional vectors $\boldsymbol{u}_i$. Then each data point $\boldsymbol{x}_n$ can be represented as:

$$\boldsymbol{x}_n = \sum_{i=1}^{D} (\boldsymbol{x}_n^T \boldsymbol{u}_i)\boldsymbol{u}_i \tag{3.11}$$

To achieve dimensionality reduction we have to restrict the number of basis vectors to be M. Therefore we can write the projected points $\tilde{\boldsymbol{x}}_n$ as:

$$\tilde{\boldsymbol{x}}_n = \sum_{i=1}^{M} z_{ni}\boldsymbol{u}_i + \sum_{i=M+1}^{D} b_i \boldsymbol{u}_i \tag{3.12}$$

Here z_{ni} depends on each datapoint and basis vector and b_i are constants. Our projection cost function is defined as:

$$J = \sum_{i=1}^{N} \|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|_2 \tag{3.13}$$

By setting the derivatives of J to zero and substituting z_{ni} and b_i we get an expression of the distortion measure dependent only on $\boldsymbol{u}_i$ (BISHOP, 2006)

$$J = \sum_{i=M+1}^{D} \boldsymbol{u}_i^T \boldsymbol{S} \boldsymbol{u}_i \tag{3.14}$$

To minimize this term we have to avoid the trivial result $\boldsymbol{u}_i = 0$, to do this we introduce the constraint $\boldsymbol{u}_i^T \boldsymbol{u}_i = 1$ leading to an modified cost function:

$$\tilde{J} = \sum_{i=M+1}^{D} \boldsymbol{u}_i^T \boldsymbol{S} \boldsymbol{u}_i + \lambda_i(1 - \boldsymbol{u}_i^T \boldsymbol{u}_i) \tag{3.15}$$

By setting the derivative to zero we can see that this terms reduces to choosing $\boldsymbol{u}_i$ to be an eigenvectors of the covariance matrix.

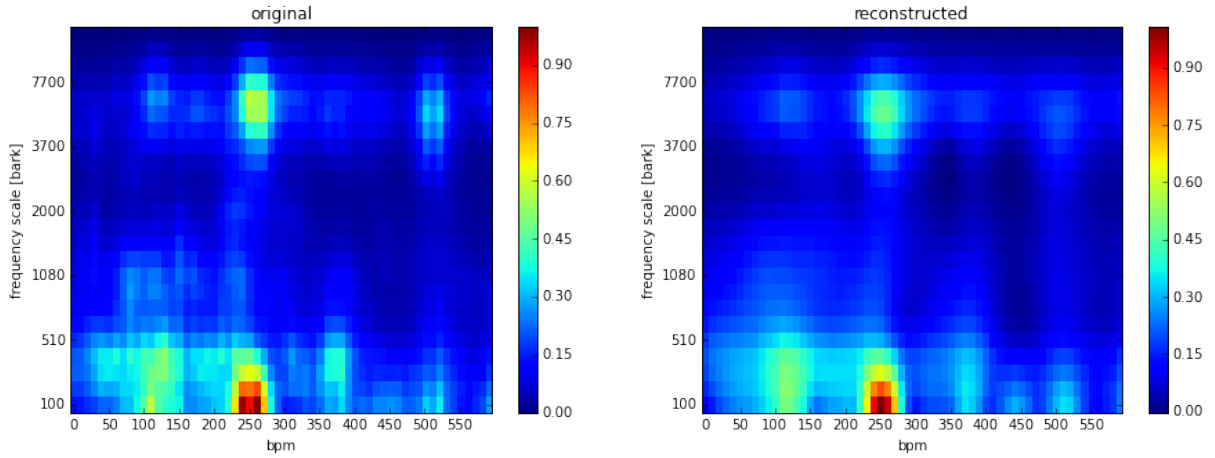
$$\boldsymbol{S}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i \tag{3.16}$$

using this observation and substituting this into 3.14 get our final projection costs function:

$$\tilde{J} = \sum_{i=M+1}^{D} \lambda_i \tag{3.17}$$

We can minimise this function by choosing the residual vectors $\boldsymbol{u}_i$, with $i = M +$

Figure 3.10: Reconstruction of compressed fluctuation pattern



$1, \ldots, D$, to be the vectors with the corresponding smallest eigenvalues. Therefore we have to choose as basis vectors the vectors with the biggest eigenvalues in $\boldsymbol{S}$. This can be done with a singular value decomposition which returns a matrix with eigenvalues on the diagonal sorted by size. To project a given vector $\boldsymbol{x}_n$ it is sufficient to multiply it with a matrix that has the M used eigenvectors of S in it's rows.

3.3.1.3 Quality of PCA

The quality of the PCA can be visualised by projecting one of the compressed vectors of dimensionality M of size 60 back to it's original dimensionality D of size 1400 and comparing the results. This has been done and visualised in Fig 3.10. It can be seen that the most important features have been preserved even though the dimensionality has been reduced by a factor of 230. The reconstructed fluctuation pattern only got a slightly smoothened out which is a desired result as it removes noise from the descriptors.

3.3.2 Expressing Music Timbre with MFCCs

Music timbre, also known as tone colour or tone quality in psychoacoustics, is what makes the same tones played with same loudness on distinct instruments sound different. In the following, three different approaches to capture dynamic evolution in music timbre are presented. Combining these approaches will lead to 4 distinct feature vectors for music timbre representation.

3.3.2.1 MFCC Deltas

Calculating the delta-cepstrals of MFCCs has been proposed in the automatic speech recognition research community by (FURUI, 1986). In hope to add dynamic information to the static MFCCs. These makes sense also for music as music evolves over time and these dynamics influence the timbre perceived by humans. It was shown by (KUMAR; KIM; STERN, 2011) that adding this dynamic component as a feature enhances the speech recognition systems it was also shown that double-delta feature further improve the system, but in this work only single delta features are used. To calculate the delta features the following formula was used:

$$\boldsymbol{D}_{mel}[n] = \boldsymbol{C}_{mel}[n+m] - \boldsymbol{C}_{mel}[n-m] \tag{3.18}$$

Where $\boldsymbol{C}_{mel}[n]$ is a single row vector at position n of the MFCC matrix. The variable m is usually in practice between 2 or 3.

3.3.2.2 Concatenated MFCCs

Another idea to represent dynamics is to concatenate multiple MFCC vectors into one vector. Describing a short changes of timbre for a musical piece. This has been done with ten such vectors leading to a feature vector of size 130. Consequently representing timbre evolution over a timeframe of $230ms$. As the double-deltas have shown to provide further improvement the same was done for the above calculated MFCC deltas. As a feature vector of 130 is still big, two more PCA over all concatenated MFCCs and Delta MFFCs were executed and the data was compressed to a dimenionality of 40.

3.3.2.3 Medians

As medians have proofed successful with Fluctuation Patterns and provide a simple and cost effective way of compressing the various vector into a single one, the medians of MFCCs as well as Deltas are computed and stored for feature combinations.

3.3.2.4 Polyphonic Timbre

It has been shown that MFCCs can be described very good with GMMs. But the methods used to compare GMMs are rather numerical unstable. To overcome this problem a polyphonic timbre representation for MFCCs was used as proposed by (YOSHII et al., 2008). These features were used in a hybrid recommender system approach and delivered very good results on

Table 3.2: Feature Vectors

Abb.	Combination	Dimensionality
A	Fluctuation Patterns Compressed + Gaussian Representation of Concatenated MFCC	90
B	Fluctuation Patterns Compressed + Concatenated MFCC Median	100
C	Fluctuation Patterns Compressed + Concatenated MFCC Delta Median	100
D	Fluctuation Patterns Compressed +(Concatenated MFCC + Concatenated MFCC Delta) Median	140
E	Fluctuation Patterns Compressed	60
F	Concatenated MFCC Medians	40
G	Concatenated MFCC Delta Median	40
H	Concatenated MFCC + Concatenated MFCC Delta Median	80
I	Gaussian Representation of Concatenated MFCC	30
J	Fluctuation Patterns Raw	1440

japanese music hits. This methods describes a songs timbre as a vector of weights for each gaussian component in a track. To calculate the polyphonic representation of timbre a IGMN (HEINEN; ENGEL, 2010) instance was trained over all concatenated MFCCs of all songs resulting in 30 components. These proofed to be quite difficult due to numerical problems caused by the dimensionality of the data. Therefore the compressed version of the concatenated MFFC version was used. Each song is then represented as 30 dimensional feature vector $\boldsymbol{u}$ describing the sum of the posterior probabilities for each vector belonging to each component:

$$\boldsymbol{u}(i) = \frac{1}{|M|} \sum_{m \in M} P(i|x_m), \quad \text{for } i = 1, \ldots, 30 \tag{3.19}$$

With m being the single feature vectors of one song and $P(i|x_m)$ being the posterior probability of x_m belonging to the i^{th} gaussian component.

3.3.3 Building Feature Vectors

From the above calculated features some combinations between them are combined to form the final track representations. The Table 3.2 illustrates how the vectors are combined, their resulting dimensionality as well as an abbreviation for them to be used subsequently.

3.4 Clustering

Given the above feature vectors it is necessary to order them into distinct clusters. This should be done considering auditory similarity of the tracks. To perform clustering the SOM algorithm was chosen as it is a sophisticated algorithm which can perform unsupervised learning and offers good ways to visualise high dimensional data. This is important as our data contains

no explicit information about similarity. Thus the algorithm used must learn by itself which tracks are similar and which are not. After training the SOM we perform a KMeans clustering on the trained codebook to further reduce the amount of clusters. We use the SOMpy implementation by (MOOSAVI, 2014), which proofed extremely fast and useful in visualisation tasks but uses heuristic which might not be optimal suited to this problem. Therefore it was tried to increase the number of training epochs with moderate success.

3.4.1 Self-organising Map Algorithm

The Self-organising Map as introduced by (KOHONEN, 2012) is neural network model inspired by certain regions of the brain which underly a 2 dimensional organization. It usually employs a two-dimensional grid of neural cells or neurons, which are all interconnected with each other. This neurons are identified by their position in two-dimensional space $\boldsymbol{r} \in A \subset \mathbb{R}^2$. Each cell is provided with the actual input $\boldsymbol{v} \in \mathbb{R}^d$ and has an weight vector $\boldsymbol{w_r} \in \mathbb{R}^d$, describing the synaptic strengths for each input element from $\boldsymbol{v}$, associated to it. A single iteration of the algorithm, in which all available data vectors $\boldsymbol{v}$ are presented to the net is described in the following steps:

1. **Initialization**

   To initialise the SOM appropriate values for all $\boldsymbol{w_r}$ have to be chosen. This can be random values if no priori information about the data is available. In this approach the principal components (see 3.3.1.2) of the data are used to initialise the weight vectors which leads to a faster convergence and therefore to lesser iterations.

2. **Choice of Input**

   A random data vector $\boldsymbol{v}$ is chosen from all available vectors to be presented to the network.

3. **Excitation Response**

   The most excited neuron is called the best matching unit or the excitation centre. To calculate a neurone excitation the euclidean distance to its weight vector is used. The neuron with the weight vector of the smallest distance to the current input $\boldsymbol{v}$ is selected as follows:

   $$\|\boldsymbol{v} - \boldsymbol{w_{r'}}\|_2 \leq \|\boldsymbol{v} - \boldsymbol{w_r}\|_2 \quad \text{,for all} \quad \boldsymbol{r} \in A \tag{3.20}$$

4. **Adaptation Step**

   This learning step is carried out by updating all neurons weight's based on a centroid

calculation between old and new synaptic strengths as shown below:

$$\boldsymbol{w}_{\boldsymbol{r}}^{new} = \boldsymbol{w}_{\boldsymbol{r}}^{old} + \eta(t) h_{\boldsymbol{rr'}}(t)(\boldsymbol{v} - \boldsymbol{w}_{\boldsymbol{r}}^{old}) \tag{3.21}$$

and continue at step 2.

With $\eta(t)$ being the hebbian learning term, a slowly decreasing function over time t as proposed by (HEBB, 2005) and used since then throughout many network models. The term $h_{\boldsymbol{rr'}}(t)$ denotes a neighbourhood topology and usually a gaussian function with decreasing radius over time is chosen. An example for this might be:

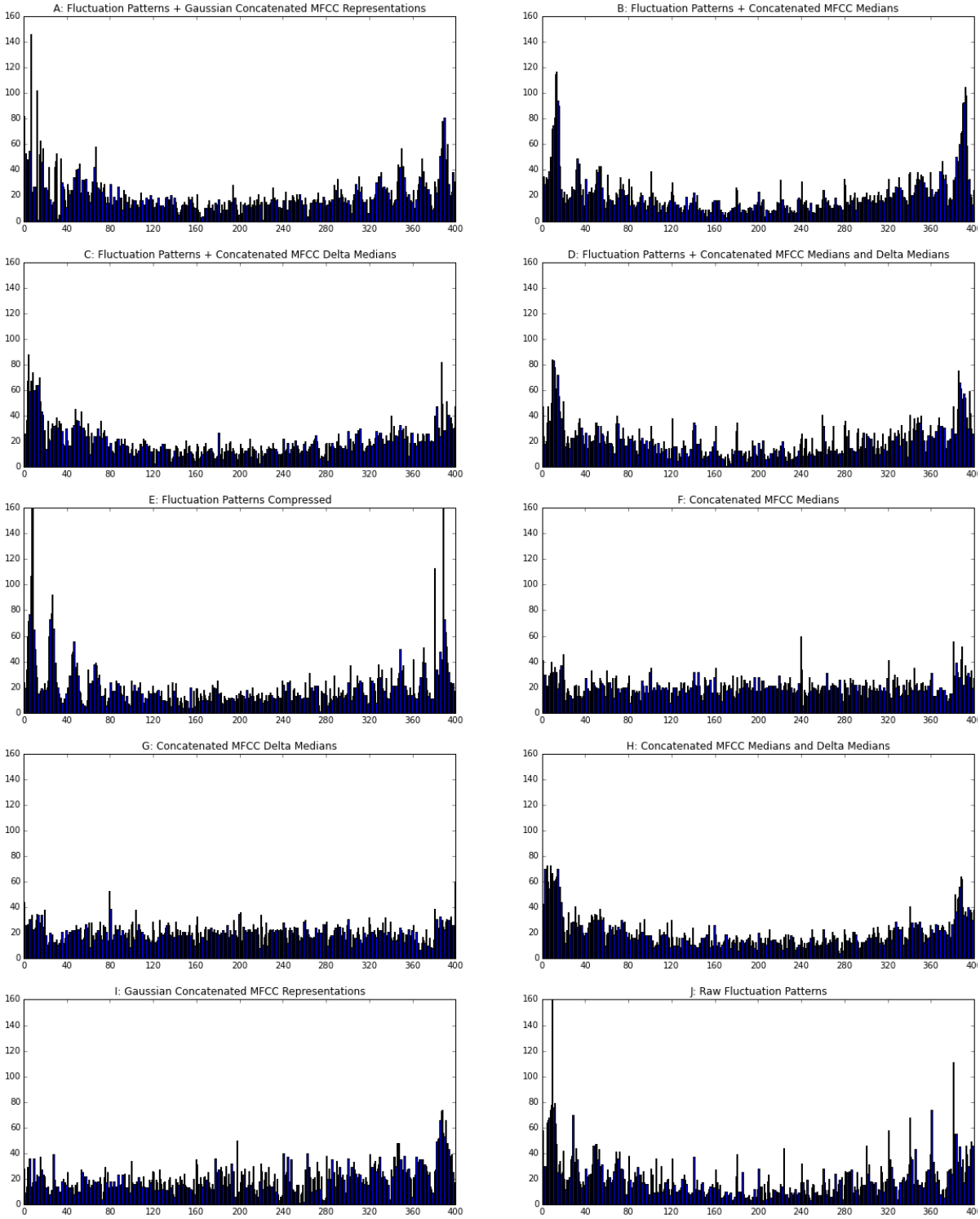
$$h_{\boldsymbol{rr'}} = e^{\frac{-(\boldsymbol{r}-\boldsymbol{r'})^2}{2\sigma_E^2}} \tag{3.22}$$

σ_E is thereby again a decreasing function over time, which determines the neighbourhood radius. It starts at rather large value σ_0 until reaching a very low value after many iterations.

3.4.2 Observations

For each feature vector as presented in Table 3.2 we initialise a SOM with a two-dimensional topology of (20×20) neurons and present all available feature vectors to it. An histogram for each SOM depicting the number of tracks associated of the 400 nodes can be seen in Figure 3.11. It can be observed that MFCC Medians features with MFCC medians tend to produce more balanced clusters whereas features using Fluctuation Patterns tend to have more variance in the cluster size. It is hard to tell which might perform better only from this results as little is known about the dataset. Considering the genre distribution there should be 3 bigger clusters and a lot of smaller ones. The plot for A ,B and D as well as F and J show the formation of some bigger clusters. To reduce the size of the clusters a K-Means algorithm is approached on the weight vectors of each some clustering our data into 40 distinct clusters. Each track is saved in a database file containing which node it belongs to for each SOM as well as to which K Means cluster it belongs to. In the next chapter we will evaluate using the dataset of user likes which features perform best.

Figure 3.11: SOM clusters using distinct feature vectors



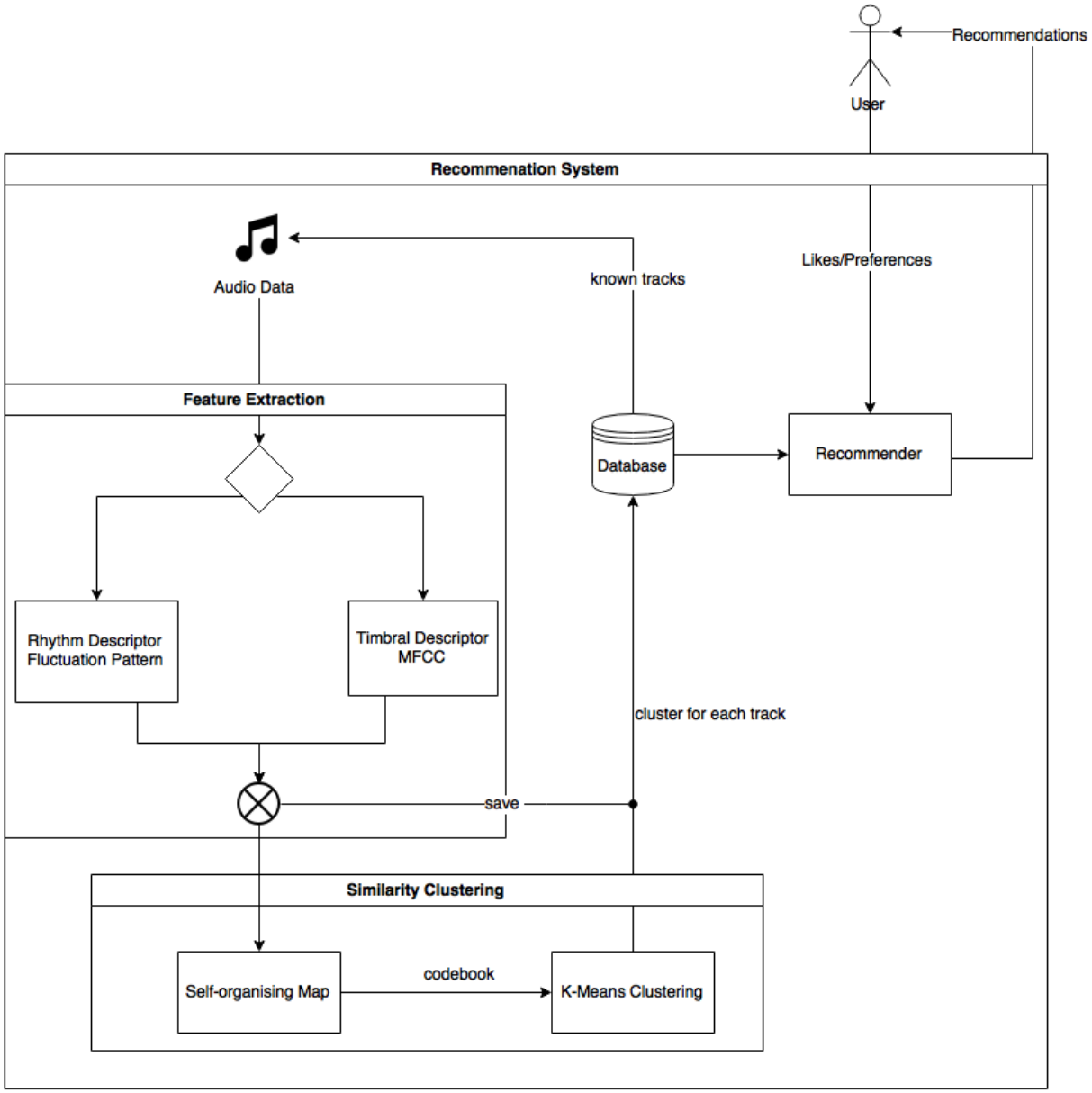
3.5 Recommender

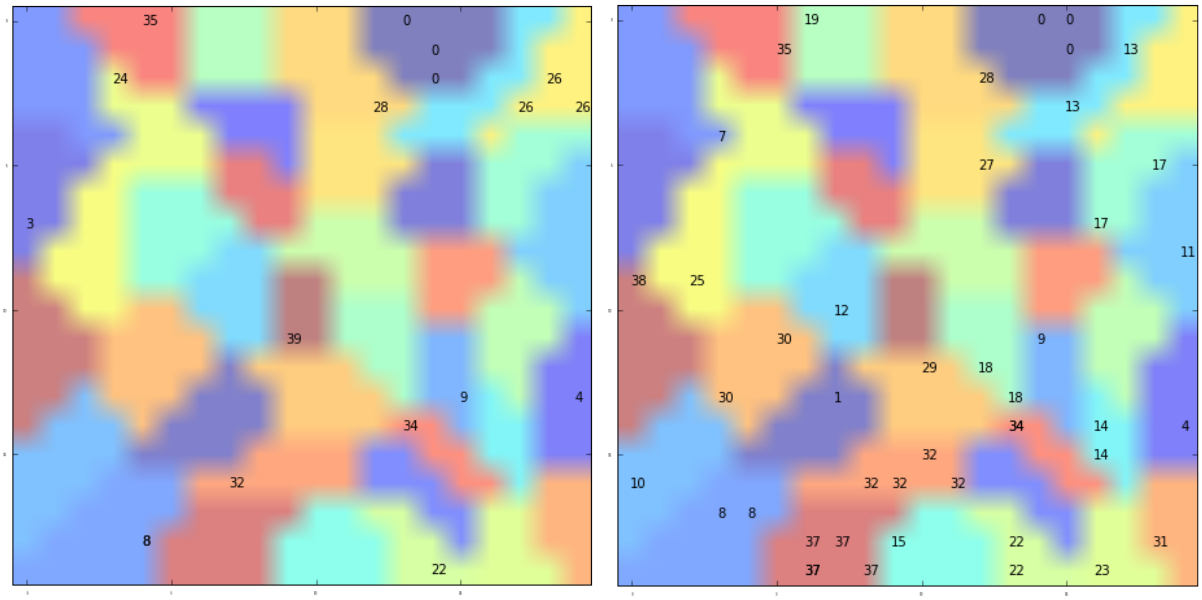
To select recommendations for a user the recommender receives a list of tracks the user has liked in the past L_u. To do this the recommender does not have to know the track before. If the track is unknown the audio is analysed and the track projected onto a cluster using the SOM algorithm like presented in the last section. To increase the neighbourhood size we perform a clustering of the codebook with the k-Means algorithm. For each liked tracks the cluster is looked up. Then the clusters are ordered by frequency, in which they appear, in the user's likes. In the following step all tracks belonging to these clusters are retrieved and presented as recommendations after removing the initial likes. Figure 3.12 shows the whole process in a simple diagram.

3.5.1 Preferences in Musical Space

In Figure 3.13 we see two different user's preferences mapped onto the k-Means clustered codebook. Each number represents a single like. It can be observed how likes are clearly concentrated in some of the clusters generated by the SOM. Thus we can conclude that similarity might be a valid approach for recommending music. It is also observable that not all likes are in clusters some likes are spread in a random manner over the map, most notably at User 1. This is congruent with humans taste as a lot of people do enjoy bigger variety of styles. Now User 1 has less likes than User 2 so it might be that this user could enjoy music from the clusters he has already likes in.

Figure 3.12: Implemented recommender system





(a) User 1 (b) User 2

Figure 3.13: Comparison of two users likes in musical space

3.5.2 Score Calculation

As one cluster usually holds a big amount of tracks a score is introduced. To calculate the score of a given recommended track t in it's cluster C_t the following formula is used:

$$\begin{aligned} s'_u(\boldsymbol{t}) &= \min(\left\{ \|\boldsymbol{l_i} - \boldsymbol{t}\|_2 \;\middle|\; l_i \in (C_t \cap L_u) \right\}) \\ s_u(\boldsymbol{t}) &= \frac{1}{s'_u(\boldsymbol{t})} \end{aligned} \tag{3.23}$$

$s'_u(\boldsymbol{t})$ presents the minimum distance of the new track to one of the users likes in the track's cluster. This distance is inverted to give a ascending score criterium with decreasing distance. This can be done for each cluster and finally the list of recommendations are sorted using the score criterium. As this step is quite computationally expensive it has been implemented to be optional as selecting tracks from a cluster usually already satisfies a certain proximity to other liked tracks in the cluster. As a solution to this dilemma a whole distance matrix between all tracks of a cluster could be pre-calculated what would reduce computational complexity. Finally the known user likes are removed from the recommended set.

4 EVALUATION

To evaluate the presented algorithm 10% of each user's likes are hidden. Then the different evaluation metrics are calculated for each user. This user scores are averaged and presented in the following tables. Furthermore for reasons of comparison we also evaluate a state of the art CF algorithm (LOW et al., 2014). Most tables also include a random recommender to provide a baseline algorithm. This random recommender randomly samples a set of tracks from all available tracks and subtracts the already known user likes.

4.1 Significance

We use the sign test to prove that our algorithm performs better than the random recommender. The sign test gives the probability that algorithm A is not truly better than algorithm B. It is counted how often Algorithm A outperforms algorithm B n_A and vice versa n_B, while $n = n_A + n_B$. We use as outperform criterium the number of recommended tracks, which appears in the evaluation set. To calculate the significance the formula 4.1 is used. To further evaluate our approach we do the sign test against the random recommender. This test is executed 10 times as the random recommenders performance might vary and the obtained results are averaged. The numbers of cluster to use was set to 40 and recommendation lists were trimmed afterwards to save computation time. Using this parameters the recommender algorithm's performance should decrease. The obtained probabilities for each feature vector can be observed in Table 4.1

$$p = 0.5^n \sum_{i=n_A}^{n} \frac{n!}{i!(n-i)!} \tag{4.1}$$

All significance levels below 0.1 have been printed bold. Usually a algorithm is considered truly better if the sign test evaluates a significance level of 0.05 or below. It can be seen that only with feature vector H and J a significance level lower than 0.05 is achieved. As the results for feature vector H are that surprising the experiments have been repeated. The test was reexecuted with 50 repetitions and the results were averaged, resulting in similar probabilities. Also feature vector A achieves sufficient significance but only at very high recommendation lengths. Feature vector D performed quite good on in Table 3.2 but never reaches less than 0.09 significance.

Table 4.1: Sign Test probabilities between our approach and random recommender

N	A	B	C	D	E	F	G	H	I	J
50	0.83	0.32	0.71	0.64	0.25	0.47	0.57	**0.07**	0.75	0.45
250	0.34	0.44	0.57	0.30	0.15	0.54	0.51	**0.02**	0.38	0.31
1000	0.41	0.18	0.23	**0.09**	**0.08**	0.41	0.59	**0.08**	0.27	0.13
2000	**0.06**	0.17	0.22	0.11	0.20	0.47	0.49	**0.09**	0.33	**0.04**

4.2 Feature Vector Evaluation

To evaluate the feature vectors we used the recall criterium, which is explained later in detail, see 4.3. For now it should be sufficient that it measures how many likes we could predict in relation to all hidden likes. We evaluated different recommendation lengths as well as number of clusters to use over all feature vectors presented in Table 3.2. The recommendation lengths are proportional to the number of clusters used $N_{recommendation} = N_{clusters} * 50$. Observe as some of them consider rhythm and timbre as for example A and D, while delivering good performance. Others like H, perform quite well but just consider timbre. Others like for example E only consider rhythm and perform very good at lower recommendation lengths but seem to weaken at very high lengths.

We choose to evaluate further metrics with H as it performed best considering significance levels. Also A will be further evaluated as it is a combination of Fluctuation patterns, which reached significance with E and J and the descriptor I which differs in it's representation from the other descriptors. Finally also D will be further considered, as it reached a low significance level and delivered best results in terms of average recall. J will not be further considered as it's use under realistic conditions is difficult because of it's high dimensionality.

4.3 Measuring Usage Prediction

As we are using unary ratings the quality of recommendation is measured by introducing metrics of how useful the recommended items are to the user. When evaluating a set of recommendations there are four possible outcomes 4.3 for the recommended and hidden items. In offline evaluation we are forced to assume that unused items would have not been used even if the have been recommended. This assumption is in most cases false as a user might use a recommended item and didn't use, or like, it because he was unaware of it's existence. The following evaluation metrics are proposed by (SHANI; GUNAWARDANA, 2011).

Table 4.2: Feature evaluation table

	Clusters used	Our Approach	CF Recommender	Random Recommender	Recommendation Size
A	1	0.87%(±4.37)	0.84%(±4.34)	0.47%(±3.62)	50
	5	3.86%(±9.36)	3.21%(±8.88)	3.28%(±8.93)	250
	20	13.89%(±17.77)	9.73%(±13.62)	12.02%(±13.96)	1000
	40	27.49%(±21.36)	18.26%(±19.34)	25.40%(±20.99)	2000
B	1	0.78%(±4.23)	0.84%(±4.34)	0.28%(±1.84)	50
	5	3.68%(±8.58)	3.21%(±8.88)	3.58%(±8.83)	250
	20	14.49%(±18.15)	9.73%(±13.62)	13.19%(±15.58)	1000
	40	26.62%(±21.08)	18.26%(±19.34)	26.82%(±21.99)	2000
C	1	0.62%(±4.19)	0.84%(±4.34)	0.58%(±3.40)	50
	5	3.72%(±9.05)	3.21%(±8.88)	2.90%(±7.79)	250
	20	13.12%(±16.16)	9.73%(±13.62)	12.94%(±16.34)	1000
	40	27.27%(±21.22)	18.26%(±19.34)	27.75%(±22.25)	2000
D	1	**1.20%(±5.82)**	0.84%(±4.34)	0.58%(±4.12)	50
	5	**4.16%(±11.04)**	3.21%(±8.88)	3.06%(±7.99)	250
	20	**17.35%(±19.28)**	9.73%(±13.62)	11.38%(±15.10)	1000
	40	**30.58%(±23.28)**	18.26%(±19.34)	24.78%(±21.42)	2000
E	1	1.12%(±4.89)	0.84%(±4.34)	0.60%(±3.11)	50
	5	3.80%(±9.17)	3.21%(±8.88)	2.93%(±9.00)	250
	20	14.36%(±16.25)	9.73%(±13.62)	12.02%(±14.59)	1000
	40	27.45%(±21.77)	18.26%(±19.34)	26.85%(±22.45)	2000
F	1	0.74%(±4.17)	0.84%(±4.34)	0.62%(±3.53)	50
	5	3.53%(±8.82)	3.21%(±8.88)	3.25%(±9.89)	250
	20	13.70%(±16.81)	9.73%(±13.62)	13.17%(±17.06)	1000
	40	25.46%(±21.67)	18.26%(±19.34)	25.73%(±21.97)	2000
G	1	0.84%(±4.24)	0.84%(±4.34)	0.73%(±4.27)	50
	5	3.75%(±9.09)	3.21%(±8.88)	3.22%(±8.95)	250
	20	11.64%(±15.65)	9.73%(±13.62)	13.27%(±18.05)	1000
	40	25.29%(±21.48)	18.26%(±19.34)	24.63%(±21.09)	2000
H	1	0.86%(±4.73)	0.84%(±4.34)	1.52%(±6.75)	50
	5	3.61%(±8.82)	3.21%(±8.88)	2.89%(±8.17)	250
	20	15.23%(±18.47)	9.73%(±13.62)	11.89%(±15.13)	1000
	40	29.74%(±22.68)	18.26%(±19.34)	24.77%(±21.52)	2000
I	1	0.61%(±3.58)	0.84%(±4.34)	0.62%(±3.56)	50
	5	3.08%(±8.79)	3.21%(±8.88)	3.37%(±9.25)	250
	20	13.05%(±17.11)	9.73%(±13.62)	12.71%(±16.47)	1000
	40	25.07%(±22.29)	18.26%(±19.34)	25.81%(±21.06)	2000
J	1	0.71%(±4.59)	0.84%(±4.34)	0.57%(±4.21)	50
	5	3.82%(±9.48)	3.21%(±8.88)	2.92%(±8.41)	250
	20	13.38%(±16.97)	9.73%(±13.62)	12.39%(±15.03)	1000
	40	26.26%(±23.12)	18.26%(±19.34)	25.67%(±21.79)	2000

Table 4.3: Outcomes for unary ratings

	Recommended	Not Recommended
Used	True Positive	False Negative
Not Used	False Positive	True Negative

Table 4.4: Usage Predictions

Feature Vector A			
N	Precision	Recall	FPR
50	$0.104\%(\pm 0.479\%)$	$0.874\%(\pm 4.371\%)$	$0.625\%(\pm 0.003\%)$
250	$0.085\%(\pm 0.178\%)$	$3.865\%(\pm 9.363\%)$	$3.123\%(\pm 0.005\%)$
1000	$0.082\%(\pm 0.105\%)$	$13.888\%(\pm 17.772\%)$	$12.494\%(\pm 0.011\%)$
2000	$0.082\%(\pm 0.078\%)$	$27.492\%(\pm 21.363\%)$	$24.988\%(\pm 0.013\%)$
Feature Vector D			
N	Precision	Recall	FPR
50	$0.104\%(\pm 0.479\%)$	$1.204\%(\pm 5.823\%)$	$0.625\%(\pm 0.003\%)$
250	$0.091\%(\pm 0.196\%)$	$4.163\%(\pm 11.044\%)$	$3.123\%(\pm 0.006\%)$
1000	$0.094\%(\pm 0.103\%)$	$17.349\%(\pm 19.277\%)$	$12.493\%(\pm 0.011\%)$
2000	$0.084\%(\pm 0.071\%)$	$30.575\%(\pm 23.279\%)$	$24.988\%(\pm 0.014\%)$
Feature Vector H			
N	Precision	Recall	FPR
50	$0.080\%(\pm 0.392\%)$	$0.864\%(\pm 4.729\%)$	$0.625\%(\pm 0.002\%)$
250	$0.078\%(\pm 0.170\%)$	$3.611\%(\pm 8.815\%)$	$3.124\%(\pm 0.005\%)$
1000	$0.086\%(\pm 0.109\%)$	$15.233\%(\pm 18.470\%)$	$12.494\%(\pm 0.012\%)$
2000	$0.085\%(\pm 0.077\%)$	$29.736\%(\pm 22.677\%)$	$24.987\%(\pm 0.014\%)$

$$\text{Precision} = \frac{|tp|}{|tp| + |fp|} \tag{4.2}$$

$$\text{Recall(True Positive Rate)} = \frac{|tp|}{|tp| + |fn|} \tag{4.3}$$

$$\text{False Positive Rate} = \frac{|fp|}{|fp| + |tn|} \tag{4.4}$$

In Table 4.4 we see these evaluation metrics on our selected feature vectors. While we can see that descriptor D performs best, the precision is verly low at all times.

4.4 Ranking Measure

Usually recommendations are presented to a user in an ordered fashion, for example in a horizontal list, with the first element being the most important one. As recommendations lists can be long it is useful to order this list according to the user's preferences. As we have "true" order given for the evaluation set of likes, only the utility of the system's ordering can be measured. The utility of each recommendation is the utility of the recommended item discounted by a factor that depends on its position in the list of recommendations(SHANI; GUNAWARDANA,

Table 4.5: RScores

Descriptor	A	D	H	Random
R-Score	0.025	0.052	**0.116**	0.085

2011). The R-Score metric assumes that a user will only see or browse the very first items of a list and the utility of recommended tracks declines exponentially with distance to the first position. Therefore the value of an item only depends on it's position in the recommendation list.

Only with descriptor H we can reach a better result than the random recommender. The random recommender results were averaged over 50 executions of the experiment. The other two descriptor seem to contain most likes very late in the list meaning that the preferences are not as close in space as with descriptor H.

$$R_u = \sum_j \frac{r_{ui_j}}{2^{\frac{j-1}{\alpha-1}}} \tag{4.5}$$

Where $r_{ui_j} = 1$ if the user u selects item i and i_j is the item at j^{th} position in the list. After obtaining all user's R-Scores the metric can be aggregated using:

$$R = 100 \frac{\sum_u R_u}{\sum_u R_u^*} \tag{4.6}$$

with R_u^* being the maximum R-Score that can be obtained by user u.

4.5 Item Space Coverage

Item space coverage refers to the proportion of items which can be recommended by the recommender. The item space coverage of our recommender is 100%. The algorithm is able to recommend each track known to him independently if there is user data available to him. Each track that is in proximity to some like of some user can and will be recommended to this user.

4.6 Observations

As seen in evaluation the algorithm presented is yet to improve and to be exposed to different datasets to give a final statement about it's performance. It showed that it has slightly superior performance to the CF algorithm at low recommendations lengths. The significance level is reached by H at a reasonable recommendation length the other feature vectors reach almost significance but only at very high recommendation lengths. Recommendation lists of this size are of low use to the real user. But it also has to be considered that offline evaluation is not an optimal method as it makes some risky assumptions about false positives. It should be also mentioned that on this dataset even the popular collaborative filtering algorithm performed rather poorly not reaching any significance at all even with long recommendation lists. This might be due to the very sparse user data that was given. We had 8003 tracks and 250 users. Therefore an matrix with over 2 million entries but only about 14000 is constructed. That results in a relative population of only 0.7%. Also the polyphonic timbre vectors similar as the ones described by (YOSHII et al., 2008) could not reproduce the results they achieved with their model based approach. This might be due to differences in the data used for analysis and evaluation or because concatenated MFCCs are not useful for this method. The data that was used to evaluate their algorithm was not available so the experiments could not be reproduced. It was seen though in 3.13 ,that many user likes form clusters in our music similarity space. Therefore it can be assumed that a certain similarity measure was achieved with this method.

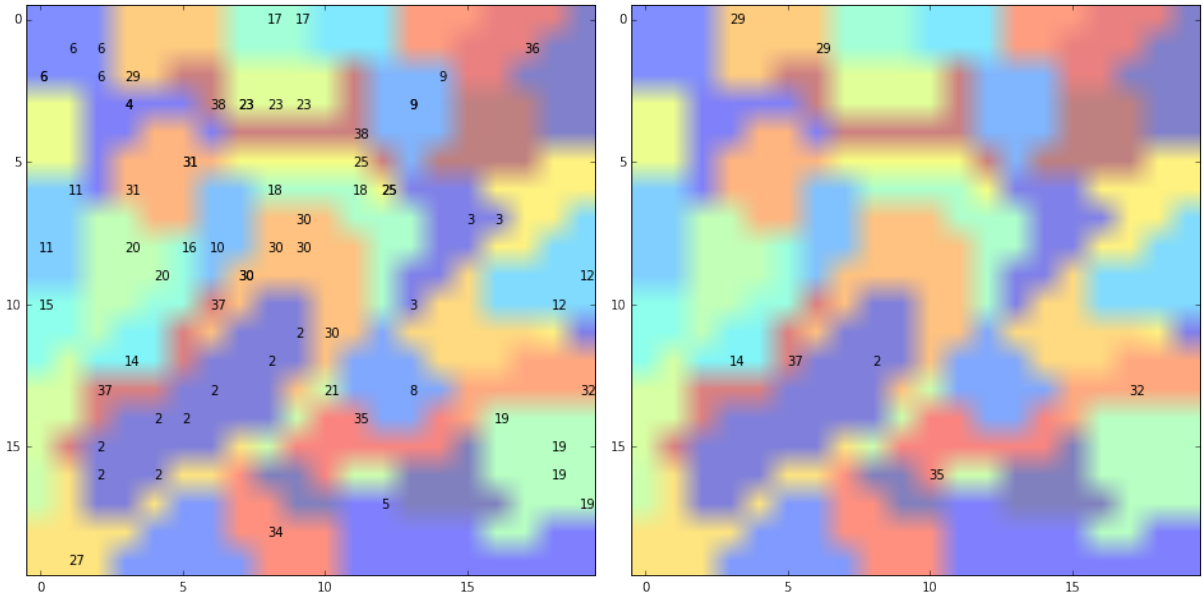
5 CONCLUSION

5.1 Summary and Discussions

A recommender system based on audible music similarity was presented. To extract rhythm features Fluctuation Patterns were calculated and to capture musical timbre MFCCs and features based on them were evaluated. This features were used to build an musical space with the SOM algorithm and a recommender was prototyped using this subspace as music similarity measure. We than evaluated the algorithm on a random chosen subset of the Million Song Dataset and compared it to a state of the art CF recommender. Significance over the baseline random recommender could be achieved but mostly at very big recommendation lengths whereas the CF algorithm could not achieve any significance on this data at all. With further adjustments this algorithm could proof useful as a baseline algorithm for systems with sparse user data as it at least performs better than a completely random recommender often used in such situations. The goal has only been reached partially as it could be shown, that likes at least seem to be influenced at least by some similarity measure. Descriptors for music analysis are yet to improve as they usually use methods from speech recognition which might be not as tailored to the problem as own feature vectors. Also the fluctuation patterns, which were developed especially as music descriptors, have actually shown to be problematic as some steps do actually worsen the result (LIDY; RAUBER, 2005). Subjective listening to single clusters and recommendation has been very interesting and it seems that descriptor D is to able span a quite good similarity space, as is A. H does certainly introduce a similarity in timbre but subjectively was found to select very dissimilar tracks considering rhythm. Evaluation should be redone on different data to confirm obtained results about significance of this descriptor.

5.2 Reviewing Data and Algorithm

Below we see a user's like set split into the two sets we used for our evaluation. This split is completely random as no time labels where available for the user data. Therefore this introduces a quite unrealistic model of user's liking data as they are more probable to like data to music they have been listening to more recently. As we see in the picture some tracks are very far away from the main clusters as for example like 11 and 33 are never going to be recommended as they are too far away from all the others and have no neighbour like in their cluster. But the ones in cluster 27 and 36 are likely to be recommended somewhere at bigger



(a) Known Likes (b) Hidden Likes

Figure 5.1: The two like sets of a user used for evaluation in music similarity space

recommendation lengths. The highest scores will be given to clusters 19 and 36 as they contain very dense likes so the algorithm will rather recommend likes from this cluster as it assumes that the user has a preference for this kind of music. Considering this information it does make sense why the significance is achieved only at high recommendations lengths.

5.3 Future Work

It stays interesting how this algorithm would perform in an online evaluation which could give concrete insights about how good the calculated musical space is representing actual music similarity even though the concept of similarity might vary drastically between distinct subjects. Then features and clustering algorithm could be adjusted by choosing the best results from online evaluation. For clustering the music space advanced methods like probabilistic methods with soft cluster or hierarchical techniques, which subdivide big clusters further, could be used to cluster this rather complex space. Furthermore it would be interesting to construct a system that learns similarity from a users listening history, so it adjust it's similarity measure to the individual.

REFERENCES

BERENZWEIG, A. et al. A large-scale evaluation of acoustic and subjective music-similarity measures. **Computer Music Journal**, MIT Press, v. 28, n. 2, p. 63–76, 2004.

BERTIN-MAHIEUX, T. et al. The million song dataset. In: **Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)**. [S.l.: s.n.], 2011.

BISHOP, C. M. **Pattern recognition and machine learning**. [S.l.]: springer, 2006.

EKSTRAND, M. D.; RIEDL, J. T.; KONSTAN, J. A. Collaborative filtering recommender systems. **Foundations and Trends in Human-Computer Interaction**, Now Publishers Inc., v. 4, n. 2, p. 81–173, 2011.

FASTL, H. Fluctuation strength and temporal masking patterns of amplitude-modulated broadband noise. **Hearing Research**, Elsevier, v. 8, n. 1, p. 59–69, 1982.

FASTL, H.; ZWICKER, E. **Psychoacoustics: Facts and models**. [S.l.]: Springer Science & Business Media, 2007.

FURUI, S. Speaker-independent isolated word recognition based on emphasized spectral dynamics. In: IEEE. **Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.** [S.l.], 1986. v. 11, p. 1991–1994.

GERNET, S. K. **Musical discrimination at various age and grade levels**. [S.l.]: The College press, 1940.

HARIRI, N.; MOBASHER, B.; BURKE, R. Context-aware music recommendation based on latenttopic sequential patterns. In: ACM. **Proceedings of the sixth ACM conference on Recommender systems**. [S.l.], 2012. p. 131–138.

HEBB, D. O. **The organization of behavior: A neuropsychological theory**. [S.l.]: Psychology Press, 2005.

HEINEN, M. R.; ENGEL, P. M. An incremental probabilistic neural network for regression and reinforcement learning tasks. In: **Artificial Neural Networks–ICANN 2010**. [S.l.]: Springer, 2010. p. 170–179.

HERLOCKER, J.; KONSTAN, J. A.; RIEDL, J. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. **Information retrieval**, Springer, v. 5, n. 4, p. 287–310, 2002.

JANSSON, A.; RAFFEL, C.; WEYDE, T. This is my jam?data dump.

KOHONEN, T. **Self-organization and associative memory**. [S.l.]: Springer, 2012.

KUMAR, K.; KIM, C.; STERN, R. M. Delta-spectral cepstral coefficients for robust speech recognition. In: IEEE. **Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on**. [S.l.], 2011. p. 4784–4787.

LASS, N. J. **Review of Speech and Hearing Sciences**. [S.l.]: Elsevier Health Sciences, 2012.

LIDY, T.; RAUBER, A. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In: **ISMIR**. [S.l.: s.n.], 2005. p. 34–41.

LOGAN, B. et al. Mel frequency cepstral coefficients for music modeling. In: **ISMIR**. [S.l.: s.n.], 2000.

LOW, Y. et al. Graphlab: A new framework for parallel machine learning. **arXiv preprint arXiv:1408.2041**, 2014.

MOOSAVI, V. **SOMpy A Self Organizing Map (SOM) Package in Python**). 2014. <http://vahidmoosavi.com/2014/02/18/a-self-organizing-map-som-package-in-python-sompy/>. Accessed: 03.12.2015.

NIEMANN, H. **Klassifikation von mustern**. [S.l.]: springer-Verlag, 2013. 209–216 p.

OHM, J.-R.; LÜKE, H. D. **Signalübertragung: Grundlagen der digitalen und analogen Nachrichtenübertragungssysteme**. [S.l.]: Springer-verlag, 2010. 56–58 p.

PAMPALK, E. **Islands of music: Analysis, organization, and visualization of music archives**. [S.l.]: na, 2001.

RAUBER, A.; PAMPALK, E.; MERKL, D. **Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by sound similarity**. [S.l.]: na, 2002.

SCHREIBER, H. Improving genre annotations for the million song dataset. In: **ISMIR**. [S.l.: s.n.], 2015. p. 241–247.

SCHROEDER, M. R.; ATAL, B. S.; HALL, J. Optimizing digital speech coders by exploiting masking properties of the human ear. **The Journal of the Acoustical Society of America**, Acoustical Society of America, v. 66, n. 6, p. 1647–1652, 1979.

SHANI, G.; GUNAWARDANA, A. Evaluating recommendation systems. In: **Recommender systems handbook**. [S.l.]: Springer, 2011. p. 257–297.

TU-WIEN, I. **MAGD Dataset MSD Allmusic Genre Dataset**. 2014. <http://www.ifs.tuwien.ac.at/mir/msd/MAGD.html>. Accessed: 03.12.2015.

TZANETAKIS, G.; COOK, P. Musical genre classification of audio signals. **Speech and Audio Processing, IEEE transactions on**, IEEE, v. 10, n. 5, p. 293–302, 2002.

UITDENBOGERD, A. L.; SCHYNDEL, R. G. van. A review of factors affecting music recommender success. In: **ISMIR**. [S.l.: s.n.], 2002. v. 2, p. 204–208.

VEMBU, S.; BAUMANN, S. A self-organizing map based knowledge discovery for music recommendation systems. In: **Computer music modeling and retrieval**. [S.l.]: Springer, 2005. p. 119–129.

WELLS, A.; TOKINOYA, H. The genre preferences of western popular music by japanese adolescents. **Popular Music & Society**, Taylor & Francis, v. 22, n. 1, p. 41–53, 1998.

YOSHII, K. et al. An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. **Audio, Speech, and Language Processing, IEEE Transactions on**, IEEE, v. 16, n. 2, p. 435–447, 2008.