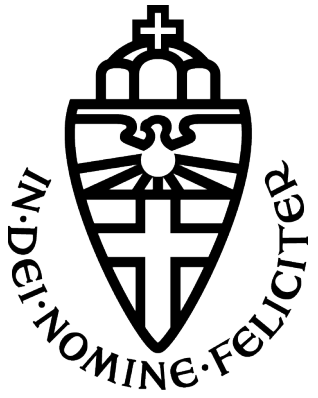
Radboud Universiteit Nijmegen



Bachelorthesis Information Science

Finding musical genre similarity using machine learning techniques

Author:
Laurens van Dijk
laurensvandijk@student.ru.nl

Supervisor:
Dr. Tom Heskes
t.heskes@science.ru.nl

July 3, 2014

In this thesis I attempt to find a quantitative measure for the similarity of musical genres. In order to find this measure, I extract features from mp3-files using MIRToolbox, classify them using LibSVM, calculate the distance between genres using the decision values of the classification process and use Matlab's agglomerative hierarchical cluster tree functionality to create a visual and numerical representation of similarity between genres. A comparison is also made to the results of clustering the raw data.

Contents

1 Introduction **4**

2 Prior research **4**

3 Motivation & approach **4**
3.1 Method . . . 4
3.2 Choice of data set . . . 5
3.2.1 Genres . . . 5
3.2.2 Songs . . . 5
3.3 Feature extraction . . . 6
3.4 Classification . . . 7
3.4.1 SVM implementation . . . 7
3.5 Hierarchical clustering . . . 8
3.5.1 Implementation and Visualisation . . . 8

4 Results and Conclusions **9**
4.1 Classification . . . 9
4.2 Genre similarity . . . 9

5 Reflection **10**
5.1 SVM accuracy . . . 10
5.2 Genre similarity . . . 10
5.3 Performance in comparison to direct clustering . . . 11

6 References **11**

7 Appendix A: Track lists **12**
7.1 Deephouse . . . 12
7.2 Electrohouse . . . 13
7.3 Dubstep . . . 14
7.4 Drum & Bass . . . 15
7.5 Hardcore . . . 16
7.6 Hardstyle . . . 17
7.7 Trance . . . 17
7.8 Rock . . . 18
7.9 Jazz . . . 19

8 Appendix B: Code **21**
8.1 processfolder.m . . . 21
8.2 processmp3.m . . . 21
8.3 automaticParameterSelection.m . . . 22
8.4 classifyUsingCrossValidation.m . . . 23
8.5 OptionCV . . . 24
8.6 svmprep.m . . . 24
8.7 svmprep2.m . . . 25

9 Appendix C: Dropped features **25**

1 Introduction

Musical genres are fascinating phenomena to study; genres can be described as a form of label, intended to order music in various categories and styles. In theory then, a genre should provide some information on how a certain style of music sounds based on certain characteristics. In practice however, what defines a genre is hugely influenced by cultural bias and industry trends.

For example, the popular on-line music store Beatport [19] lists five different 'flavours' of house music. To an untrained ear, it is quite unclear what the difference between these five genres is.

To complicate matters even further, the definition of a musical genre can very well change over time: Rock songs made seventy years ago are radically different from rock songs today, and with modern genres this change factor is even more pronounced.

Another factor that complicates studying genres is that genres can fall into arbitrary hierarchies of genres: Some genres could be said to be quite closely related, such as the aforementioned flavours of house music. Other genres define a group of subgenres, or are radically different from most other music.

McKay and Fujinaga make a distinction relevant to these hierarchies in their widely-used data set [5] between so called 'root' genres and 'leaf' genres, where a root genre can either be a solitary genre or a grouping genre for several leaf- and/or root-genres.

When this concept of genre types is applied to the five house genres mentioned earlier, these 'flavours' of house could then be seen as leaf genres falling underneath a certain root-genre. As a root genre for these leaf genres, a likely candidate seems to be 'EDM', or Electronic Dance Music.

It is this distinction between genres, and the relation between sets of genres that seemed like an interesting concept to explore for my bachelor thesis.

In particular, I am very curious to see whether a certain quantitative measure can be found for the amount of similarity between arbitrary genres. In this thesis, I will be combining various elements of previous research and my own work in order to find a quantitative measure, with emphasis placed on the use of a classifier in order to generate data that can be used to determine the similarity between genres.

2 Prior research

Some research in the field of machine-learning approaches to audio problems are of particular significance with respect to this thesis. C. Silla and A. Freitas compare several approaches in order to create a musical hierarchy in their research [1]. One of the approaches used in this paper is an SVM-based classifier, which provided useful information on the feasability of using an SVM-based approach for my own study.

With the choice for an SVM-based approach in mind, the team of S. Brecheisen also evaluated the performance of an SVM-based approach in their work on the creation of a framework for musical hierarchy generation [2]. However, their focus was more on large scale classification than on individual distances.

Another important study is that of Tzanetakis and Cook [6], who were one of the earlier research groups to attempt genre classification. In this particular study, the researchers use musical features in order to classify various songs as belonging to a certain genre. The use of these features provided me with the information that using musical features is a good way to differentiate between various genres, in particular when classifying songs as belonging to these genres.

In this thesis, I will try to build upon this earlier research and try to test the viability of a new way of defining the distance between arbitrary and closely related genres using tested classification methods.

3 Motivation & approach

3.1 Method

The process of finding a quantitative measure for musical genre similarity can be broken down into four chunks: First, a data set has to be constructed containing songs belonging to several genres. These songs then have to be analysed, and mathematical features have to be extracted. We need these features for the next step, using a technique to define the distance between various genres. Finally, the defined distances have to be evaluated and if possible visually displayed.

In the next few chapters I will discuss my method in detail. There is, however, one step that needs some explaining at this point, and that is the method

of defining the distance between various genres. A possible conventional approach would be to use some kind of clustering technique directly on the extracted features, and using the output of this technique to compare various genres. For the purpose of this thesis, however, I wanted to find out if a familiar classification technique could be used to find if various musical genres are similar to each other.

Instead of clustering the raw features directly, I have replaced this process by first training a classifier using the extracted features and given ground truth labels. The output of this classification process is then used to determine the distance between various genres. The basic idea behind this process is that genres receiving similar classification scores for a series of observations are probably similar to each other, and genres receiving highly different scores are probably not similar.

Evaluating whether this unconventional step outputs a credible and useful result will be done in the final chapters of this thesis.

3.2 Choice of data set

3.2.1 Genres

As stated in the introduction to this thesis, a huge, if not indeterminable number of genres exist. This implies that it would be highly unfeasible to find example songs of all musical genres. As such, I have chosen to use a subset containing a mixture of root genres and leaf genres belonging to the electronic dance music (EDM) root genre. In his paper on electronic/dance music, K. McLeod states EDM to be a relatively vaguely defined genre [8]. On the defining characteristics of EDM, McLeod states (p. 60) that EDM is:

> "[..] A heterogeneous group of musics made with computers and electronic instruments often for the purpose of dancing."

The heavy use of computers and electronic instruments is in accordance with these genres. As such, even genres not explicitly named by McLeod in his list (p. 60) could be said to belong to EDM.

In addition to the EDM leaf genres, Rock and Jazz were added as root genres in order to complete the list with genres dissimilar to EDM. The following list of genres was used:

- Deephouse
- Electrohouse
- Dubstep
- Drum & bass
- Trance
- Hardcore
- Hardstyle
- Rock
- Jazz

In this list, Rock and Jazz are both root genres. The other genres can all be classified as leaf genres of EDM.

Note that 'Hardstyle' is merely a different name for 'Hard Dance'. Both names are used by producers and retailers, with 'Hardstyle' being more common in the Netherlands. For example, Beatport uses the term 'Hard Dance' [20] while Dutch radio station Q-Dance uses the term 'Hardstyle'. [21]

Additionally, all of the genres chosen could in theory be seen as parent genres of an additional 'level' of subgenres. These 'subsubgenres' become progressively less defined, and songs belonging to these genres become progressively harder to find. I decided that for finding similarity between genres, the 'first-order' leaf-genres selected should suffice within the scope of this thesis.

It should however be mentioned that even though these 'subsub(sub, etc)genres' are not used for this thesis as a result of their being poorly defined, they do exist and are recognised by several researchers and experts.

3.2.2 Songs

Once I had a list of genres, I had to find several sets of songs belonging to a certain genre. I wanted to find between 30-50 songs to analyse for each genre. Although several excellent databased containing song data exist, it was impossible to find a data set that contained recent audio files for the EDM leaf genres. In order to have a reliable set of data including songs from these genres, I had to look for alternative sources.

The easiest way to obtain these songs turned out to be using collections, lists and/or compilations of 'top x' songs for a genre. These collections or lists consist of between 29-100 mp3-files of songs belonging to a given genre. In order to remove/prevent selection bias, I chose to use collections and lists compiled by external sources: music labels or -experts.

For an extra validation check of the chosen list's genre, I selected a random sample of five songs per

list and compared them with the genre specified for the selected songs at a second external source. For the EDM leaf genres Beatport was used, while Rock and Jazz songs were compared with their listing on the popular music scrobbling site Last.fm [22]. This choice is due to EDM genres being commonly grouped as 'electronic dance' on last.fm (without the crucial subgenre information), and due to the absence of Rock and Jazz songs on Beatport.

The following lists were used:

- Deephouse: "Ministry Of Sound - The Sound Of Deep House (2013)"
- Electrohouse: "VA - 30 Amazing Electro House Tracks (2011)"
- Dubstep: "UKF Dubstep" (50 top rated songs)
- Drum & bass: "Drum And Bass - Arena Evolution"
- Trance: "Trance Top 40 - Best Of 2013"
- Hardcore: "FearFM Hardcore Top 100 (2012)"
- Hardstyle: "Q-Dance Presents Hardstyle Top 40 (February 2014)"
- Rock: "Top 40 - Rock (2014)"
- Jazz: "100 Gold Jazz hits"

The full contents of these collections can be found in Appendix A.

3.3 Feature extraction

In order to be able to use classification and clustering techniques, a way of numerically representing features of arbitrary songs is needed. These features are musical characteristics of a song, such as tempo, pitch and key. To accomplish this, I used the musical analysis toolbox 'MIRToolbox' [18]. MIRToolbox contains a number of Matlab functions designed to extract various musical features from mp3- and wave-files and represent these as numerical data.

Three features that makes using MIRToolbox particularly useful are the ability to use mp3-files as input, casting audio-files to mono and the ability to analyse songs with a certain down-sampling factor. These features are useful since they reduce the amount of data being analysed by a large amount. For example; "Britney Spears - I Wanna Go (Martin & Souza Club Mix)", in the electrohouse track list, is 7 minutes and 49 seconds long with a bit rate of 320 kb/s and is encoded in stereo.

Without down-sampling, this would mean that all calculations performed using the MIRToolbox have to be performed on 17.8 MB of data. By summing the two audio tracks contained in the mp3-file to one mono track and down-sampling with a factor of four, the amount of data being analysed is reduced to an eighth of the original 17.8 MB, greatly speeding up the process of feature extraction. The down-sampling factor of four, or dropping three out of four frames, appears to be an appropriate balance between reducing the amount of data that has to be processed and losing too much data for reliable classification.

The following procedure was used to find the features for the songs selected in 3.2.2:

First, the first 40 songs in a collection (or all songs if the number of songs ≤ 40) are moved to a separate folder labelled with the genre name. The Matlab script `processfolder.m` is then called with this folder as input parameter. This script iterates through all .mp3 files in the folder, processing each file using `processmp3.m`. `processmp3.m` loads an mp3-file, and processes it using the aforementioned down-sampling and channel reduction options. The resulting features are saved as a tab-delimited file with `features_'songname'.dat` as filename pattern.

Once an entire folder/genre has been processed, the resulting .dat files are moved back to the original directory. The .dat files for each genre are then concatenated and saved as `'genrename'_consolidated.dat`. Finally, the consolidated files per genre are again concatenated and stored as `datafile.dat` in a separate folder. This file is the data set used for the following steps. In this data file, row 1 contains column names, the rest of the rows contain observations (songs) and the columns contain the extracted features.

In the final data set, some features were unusable for the next part of this thesis: These features would either consist solely of 'NaN'-values, had missing ('NaN') values for certain observations or had a huge disparity between values for certain observations. In order to prevent these features from greatly skewing further classification they were removed from the final data set. These features are listed in appendix C 9.

Finally, the data set is loaded in Matlab and some variables are initialised for the following steps. In addition, the feature scores are normalised using a z-score loop. Code for this variable initialisation and normalisation process can be found in 8.6.

3.4 Classification

Now that a suitable data set has been generated and converted into an easily workable format, the next step is training a classifier to distinguish between genres. The reason for doing this is that a probabilistic classifier should give a clear and objective measure of the chance of a certain number belonging to a certain genre.

For this classifier, I decided on using a support vector machine based system. This choice was based on two factors: My own experience in working with SVM-based classification systems, and the success of various research groups in using SVM's for musical classification tasks. One such group is the team of S. Brecheisen, H. Kriegel, P. Kunath and A. Pryakhin, who use a multi-layer SVM in their own research on musical genre taxonomies [2].

One problem with using SVM's is that a single SVM can only be used to solve binary classification problems: SVM's are trained on negative and positive examples of a certain classification task. The SVM then creates a best-fitting hyperplane dividing the negative and positive examples. When a new sample is then classified with a trained SVM it can only be classified as either a negative or positive example for the given task, based on the separating hyperplane. However, when viewing musical genres as nominally typed categorisations for music tracks it becomes possible to approach genre classification as a multi-class classification problem.

An advantage of using SVM's is that they can be adapted to return a probability value, based on for example the distance of an observation to the separating hyperplane. The greater the distance, the higher the chance of an observation being correctly classified.

For genre classification, this probability value can then be used to determine how representative a certain song is for a certain genre. For example, a song might return a decision value of 0.9 for genre 1, 0.7 for genre 2 and -0.8 for genre three. Using a winner-takes-all metric, the song would then be classified as belonging in genre 1, while it could also be said to be (partially) representative of genre 2.

In this thesis, I chose to use the existing open-source SVM package LibSVM [12] and adapt it to the classification task of classifying music according to genres. The latest distributions of LibSVM contain Matlab scripts that can be used to solve multi-class problems.

Multi-class problems can be approached in two ways: One possibility is creating one SVM per genre: All tracks belonging to that genre are positive samples, all other tracks are negative samples. This approach is called one-vs-all. The second possibility is creating an SVM for each combination of genres, and 'voting' on the predicted genre using the SVM's outcomes. The genre with the highest number of 'votes' is then assumed to be correct. This approach is called one-vs-one.

LibSVM uses a one-vs-one approach by default, although wrappers do exist for one-vs-all LibSVM implementations. The decision to use a one-vs-one approach was based on by C. Hsu and C. Lin's work on multi-class SVM approaches [16].

3.4.1 SVM implementation

In order to prevent overfitting, the data set is first randomly split using Matlab's `dividerand` function. When called as `[valSet,null,testSet] = dividerand(datafileD',0.3,0.0,0.7);` the data file is randomly split in 30% training/validation data and 70% test data. For the data set used in this thesis, this results in 245 test observations and 105 validation observations. The validation set is used to find optimal parameters for the SVM-classifier, while the test set is for actual classification. During classification, the test set is internally split in training and test data. Some further preparation of the training- and test data is then done using a second preparation script listed in 8.7.

As stated above, the validation data set can be used to find optimal parameters for SVM training. This is done using 10-fold crossvalidation, using a script listed in 8.3 extending LibSVM. In this script, folds are continuously tested and validated with varying parameters for LibSVM's training function. To find the parameters, this script is called as `automaticParameterSelection(valSetLabels, valSet, 10, optionCV)` with `optionCV` containing parameters for the SVM and crossvalidation function. These parameters are listed in appendix b 8.5.

Once a set of optimal parameters is found, these are appended to a string used for defining LibSVM parameters. These parameters define the kernel, gamma and cost used, among others. For the classification problem posed, the LibSVM documentation advises using a C-SVC SVM (parameter -s) with a Radial Basis Function kernel (parameter -t). The choice for an RBF-kernel was motivated by the "guide to SVM classification" [14]. In this

guide, the use of an RBF-kernel is highly recommended.

The parameter string is then defined as `opts = '-s 0 -t 2 -c [cost] -g [gamma] -b 1';` , with cost and gamma as output of the parameter selection script. The final parameter '-b' is set to 1 in order to have LibSVM output probability/decision values, which will be used in the following steps. These parameters are also copied to the `OptionCV` struct used later.

With the selected parameters, we can now attempt classification of the test set. In order to obtain a high degree of accuracy without overfitting, I used a highly modified version of an existing matlab script. This script was originally based on a one-vs-all approach, without incorporating probability values. The modified code can be found in 8.4, and outputs a series of decision values (probabilities) and the predicted labels for the last fold processed.

Indices for crossvalidation are generated using `Indices = crossvalind('Kfold', 245, 5)`, with the number of records in the test set as the second parameter. The classification script is called using `classifyUsingCrossValidation( testSetLabels,testSet,Indices,5,optionCV);`

After classification, two new variables have also been created in the workspace, containing the decision values and predicted labels mentioned earlier. The matrix containing decision values contains nine columns (for nine genres), with each column containing the 'chance' of a given observation belonging to a genre.

3.5 Hierarchical clustering

With a fully trained SVM and the decision values and predicted labels created in 3.4.1 it is now possible to try and create a measure of genre similarity. In order to do this, I will interpret the decision values as a measure of how related genres are. For example, in the decision value table the following decision values can be found for the first observation, after column names are labeled with the associated genres.

- 'deephouse' : 0,199820470440789
- 'dubstep' : 0,0578549429985571
- 'drumandbass' : 0,0804897211132723
- 'electrohouse' : 0,411321119131090
- 'trance' : 0,147730081730698
- 'hardstyle' : 0,0254642432073126
- 'hardcore' : 0,0297628505160296
- 'jazz' : 0,0354835858205612
- 'rock' : 0,0309709855543124

The above observation will result in the song being classified as electro house, since it has the highest probability. We can however also take the other decision values into account: If the song could not be classified as electro house it would be classified as the genre with the second-highest probability. In this case, somewhat unsurprisingly, deep house.

My hypothesis is that the difference and/or similarity between the decision values of a certain genre when compared to other genres can be used to define the distance between these genres: If all observations of one genre are consistently also highly ranked as a certain different genre, these two genres apparently share some overlap of musical features and/or characteristics.

This hypothesis brings us to the core of this thesis: Finding a way to use these decision values to calculate the similarity between various genres. In order to do this, I will view each column in the decision value matrix as a vector specifying the distance between a genre and the other genres. If the distance between these vectors is calculated and the genres clustered using these distances, this should result in a clear overview of potential similarity between genres.

As a distance measure, my supervisor suggested using Matlab's `pdist`-functionality combined with either a euclidean- or a cosine measure of distance. A study into scoring musical similarity [11] suggests both are valid measures of musical similarity. After evaluating both, I decided to use the cosine measure. Using this measure, the distance between objects is calculated as one minus the cosine of the included angle between points.

3.5.1 Implementation and Visualisation

With the distance measure decided upon, I could use Matlab's `pdist`-function to calculate the distance between genres in the test set. In order to get readable results, `pdist` was called using `squareform`, which converts vectors outputted by `pdist` into a 9-by-9 matrix comparing the various genres. After adding genre labels and colouring the matrix for readability, the comparison table listed in figure 1 is produced. In the next few chapters, this table will be referred to and interpreted.

When we want to create a hierarchical clustering of genres, Matlab offers a solution in the

	dph	dub	dnb	elh	trance	hst	hac	jazz	rock
dph	0,0000	0,8460	0,5886	0,3903	0,8271	0,9082	0,9200	0,9237	0,9034
dub	0,8460	0,0000	0,5659	0,8617	0,8638	0,8645	0,9312	0,9615	0,9271
dnb	0,5886	0,5659	0,0000	0,6530	0,6494	0,8002	0,8411	0,9311	0,8417
elh	0,3903	0,8617	0,6530	0,0000	0,7640	0,8946	0,9297	0,9202	0,9146
trance	0,8271	0,8638	0,6494	0,7640	0,0000	0,8005	0,8173	0,9352	0,8248
hst	0,9082	0,8645	0,8002	0,8946	0,8005	0,0000	0,8893	0,9765	0,9221
hac	0,9200	0,9312	0,8411	0,9297	0,8173	0,8893	0,0000	0,9299	0,8247
jazz	0,9237	0,9615	0,9311	0,9202	0,9352	0,9765	0,9299	0,0000	0,8331
rock	0,9034	0,9271	0,8417	0,9146	0,8248	0,9221	0,8247	0,8331	0,0000

Figure 1: Genre Matrix

form of the `linkage`-function. This function creates an agglomerative hierarchical cluster tree based on the distance between genre vectors. When called as `link = linkage(dec_values' ,'complete','cosine');` clusters are formed based on the furthest distance between clusters, using the cosine distance measure.

We can then visualise these distances using Matlab's dendrogram function `dendro`, in order to obtain the dendrograph listed in figure 2.

4 Results and Conclusions

With a fully trained SVM created, and the distance between genres calculated, it is now possible to interpret the data and conclude whether the complete system can be used as a measure of similarity between genres.

4.1 Classification

The SVM performed better than expected. In C. Silla and A. Freitas' work on musical classification an accuracy of at best 76% was obtained [1, page 5]. This is on par with trained human listeners, as shown in research done by G. Tzanetakis and P. Cook [6, page 301], in which college students managed to classify 'root' genres with around 70% accuracy. It could then be stated that the SVM trained for this study performs at least on par with human listeners, with some caveats reflected upon in 5.

4.2 Genre similarity

As a quantitative and visual similarity measure, the use of decision values as input for a distance measurement function seems to be a viable choice. As visually demonstrated in both figures 1 and 2. In the genre matrix, the distance between genres as measured using the cosine distance is displayed. This distance is given per genre, with a score of 0 implying two genres are identical. The diagonal contains a genre compared with itself, the distance of which of course always 0.

In this matrix some interesting information can be found. If we interpret the distances between genres, we can see that deep house (dph) and electrohouse (elh) appear to be relatively similar with a distance of 0.3903, for example. This could be expected as the two genres are both types of house music. Jazz, on the other hand, appears to be less similar to rock, with a larger distance of 0.8331. This is also credible, as Jazz is a root genre, implying it should bear little similarity to leaf genres of a different root genre.

It is also interesting to see that hardstyle (hst) and hardcore (hac) both bear little similarity to deep house and electrohouse, as evident in the relatively large distances varying between 0.8946 and 0.9312. These four genres are all leaf genres of the EDM root genre, so a large distance between these leaf genres can at first seem unlikely. However, I would argue that this is because these genres are determined more by rapid cultural changes than by musical similarity: All EDM leaf genres are simply grouped under EDM because of the heavy use of some EDM-specific instruments and beat patterns, as referenced in 3.2.1. To an average listener, these leaf genres can in fact not sound alike at all. This is also pointed out by K. McLeod, who states that

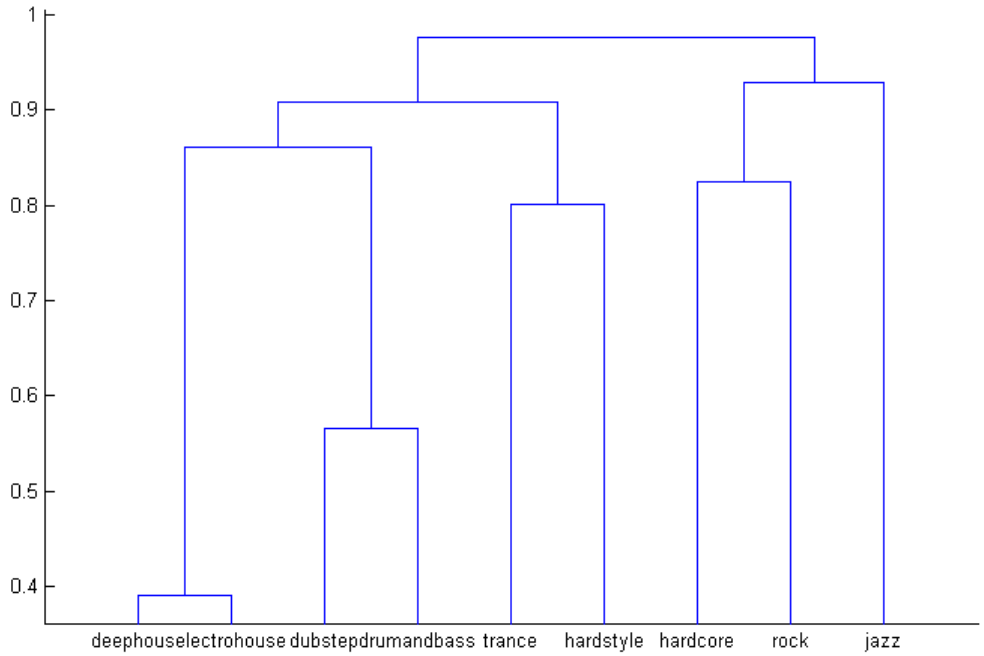


Figure 2: Genre Dendrogram

EDM is a genre label that is deliberately kept relatively vague in order to accommodate different styles [8, page 60].

When we observe the dendrogram, we find several interesting clusters supported by the distances given in the genre matrix. For example, the split deep house/electrohouse tree is unsurprising. The fact that hardcore forms a tree with jazz and rock instead of with the other EDM leaf genres is slightly surprising as well, but seems likely based on the distances listed in the genre matrix.

In conclusion, I would state that the method used in this thesis is a viable way of measuring the similarity between various musical genres. I do have some points for improvement and further research, however, which I will discuss in the following chapter.

5 Reflection

5.1 SVM accuracy

Several factors have to be taken into account when evaluating the performance of the SVM used, with the SVM's classification accuracy as a measure. The use of entire songs instead of smaller samples for feature extraction and the use of automated parameter selection techniques are such factors, as is the use of a certain SVM kernel.

The generation of data sets is also an important factor: The data set is randomly split in training- and test sets and cross validation techniques were used to test created models. As a separate (manual) check, several other runs were performed with different data splits and different numbers of cross validation folds. The final accuracy of these runs varied between 70% and 84.8%.

When compared to accuracy rates listed in C. McKay and I. Fujinaga's study [5], this level of accuracy appears to be on par with leading classification techniques.

5.2 Genre similarity

As stated in 4.2, the similarity between 'EDM' leaf-genres was not as pronounced as would be expected from genres that are leaves of an average root-genre. Even though this difference can be explained, it is still an indication that further research could be done using more 'traditional' genres.

	dph	dub	dnb	elh	trance	hst	hac	jazz	rock
dph	0,0000	1,2706	0,8495	0,3180	1,1261	1,2774	1,4935	0,9879	1,3714
dub	1,2706	0,0000	0,5116	1,1158	1,1118	0,7282	1,1395	1,5297	1,3088
dnb	0,8495	0,5116	0,0000	0,8874	0,9077	0,8137	1,1814	1,5633	1,5831
elh	0,3180	1,1158	0,8874	0,0000	0,9161	1,0972	1,4837	1,1594	1,4868
trance	1,1261	1,1118	0,9077	0,9161	0,0000	0,7802	0,8641	1,3981	1,2440
hst	1,2774	0,7282	0,8137	1,0972	0,7802	0,0000	0,9358	1,5830	1,3903
hac	1,4935	1,1395	1,1814	1,4837	0,8641	0,9358	0,0000	1,1553	0,7885
jazz	0,9879	1,5297	1,5633	1,1594	1,3981	1,5830	1,1553	0,0000	0,5404
rock	1,3714	1,3088	1,5831	1,4868	1,2440	1,3903	0,7885	0,5404	0,0000

Figure 3: Genre matrix based on raw data

For instance, the SAC data set mentioned by C. McKay and I. Fujinaga [5] contains 'Classical', 'Jazz', 'Rap', 'Blues' and 'Rock' as genres. It would be very interesting to see how these genres perform in relation to the genres chosen for this study, while using the same methods. However, due to time and resource constraints finding and analysing songs using the same methods as used for the currently analysed genres will have to be filed as an interesting avenue for future research.

5.3 Performance in comparison to direct clustering

Even though the results of the process used in this thesis show that the describes process of defining the distance between genres is a feasible method, I haven't done a full comparison with the results of skipping the classification step entirely and using the raw data as basis for distance calculations.

A possible and unvalidated way of doing this would be to calculate the average value for each feature per genre and storing the results in a 9-by-91 matrix 'D'. With `pdist(D,'cosine')`, it is then possible to calculate the distances between the nine genres, based on the average values of all the features. Using the data set containing all 350 songs, this results in the graph listed in figure 3.

At first sight, both the results after classification as well as the results based on direct distance calculations could also be used to determine the distance between various genres. However, the distances appear to scattered more, and it is interesting to see that values greater than 1 are found.

A follow-up study in whether the differences in distances after classification when compared to the raw data matrix significantly alter the results would be quite interesting.

6 References

[1] Silla, C. N., & Freitas, A. A. (2009, October). Novel top-down approaches for hierarchical classification and their application to automatic music genre classification. In Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on (pp. 3499-3504). IEEE.

[2] Brecheisen, S., Kriegel, H. P., Kunath, P., & Pryakhin, A. (2006, July). Hierarchical genre classification for large music collections. In Multimedia and Expo, 2006 IEEE International Conference on (pp. 1385-1388). IEEE.

[3] Burred, J. J. (2005). A hierarchical music genre classifier based on user-defined taxonomies. In Proc. ISMIR.

[4] Cesa-Bianchi, N., Gentile, C., & Zaniboni, L. (2006, June). Hierarchical classification: combining Bayes with SVM. In Proceedings of the 23rd international conference on Machine learning (pp. 177-184). ACM.

[5] McKay, C., & Fujinaga, I. (2010, March). Improving automatic music classification performance by extracting features from different types of data. In *Proceedings of the international conference on Multimedia information retrieval* (pp. 257-266). ACM.

[6] Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *Speech and Audio Processing, IEEE transactions on, 10(5)*, 293-302.

[7] Pachet, F., Westermann, G., & Laigre, D. (2001, November). Musical data mining for electronic music distribution. In Web Delivering of Music, 2001. Proceedings. First International Conference on (pp. 101-106). IEEE.

[8] McLeod, K. (2001). Genres, subgenres, sub-subgenres and more: Musical and social differentiation within electronic/dance music communities. Journal of Popular Music Studies, 13(1), 59-75.

[9] Chen, Y., Crawford, M. M., & Ghosh, J. (2004, September). Integrating support vector machines in a hierarchical output space decomposition framework. In Geoscience and Remote Sensing Symposium, 2004. IGARSS'04. Proceedings. 2004 IEEE International (Vol. 2, pp. 949-952). IEEE.

[10] K. Kampa (n.d.). Kittipat's Homepage - https://sites.google.com/site/kittipat/libsvm_matlab

[11] Seo, J., Park, N., & Lee, S. (2012). A music similarity function based on the Fisher kernels. Proc. CMMR, 429-436.

[12] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

[13] Huang, A. (2008, April). Similarity measures for text document clustering. In Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand (pp. 49-56).

[14] Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification.

[15] Scheirer, E. D. (1998). Tempo and beat analysis of acoustic musical signals. The Journal of the Acoustical Society of America, 103(1), 588-601.

[16] Hsu, C. W., & Lin, C. J. (2002). A comparison of methods for multiclass support vector machines. Neural Networks, IEEE Transactions on, 13(2), 415-425.

[17] Hunt, M., Lennig, M., & Mermelstein, P. (1980, April). Experiments in syllable-based recognition of continuous speech. In Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'80. (Vol. 5, pp. 880-883). IEEE.

[18] https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox

[19] Beatport. (n.d.). About us. In *Beatport Support Center*. Retrieved June 20th, 2013, from https://knowledgebase.beatport.com/kb/article/000116.

[20] Beatport. (n.d.). Hard-dance. Retrieved May 29th, 2014, from http://www.beatport.com/genre/hard-dance/8

[21] Q-Dance. (n.d.) Hardstyle Top 40. Retrieved May 29th, 2014, from http://www.hardstyletop40.com/

[22] Last.FM. (n.d.) "Discover more music" Retrieved May 29th, 2014, from http://www.last.fm/

7 Appendix A: Track lists

7.1 Deephouse

- AlunaGeorge - Your Drums, Your Love (Duke Dumont Remix)
- Cloud 9 - Do You Want Me Baby (Dusky Remix)
- Lee Foss & MK - Electricity (Feat. Anabel)

- Storm Queen - Look Right Through (MK Dub III)
- Toyboy & Robin - Jaded
- Tensnake - Mainline (Feat. Syron)
- Artful Dodger - Please Don't Turn Me On (Feat. Lifford - Disclosure Remix)
- Osunlade - Envision (Argy Vocal Mix)
- Dusky - Flo Jam
- Yousef - Beg (Hot Since 82 Future Mix)
- Shadow Child - String Thing
- Hot Since 82 - Knee Deep In Louise
- George Fitzgerald - Child
- Noir & Haze - Around (Subb-an Remix)
- Bicep - Vision Of Love
- Flashmob - Need In Me
- FCL - Its You (San Sodas Panorama Bar Acca Version)
- Intruder - Amame (Feat. Jei - Long Ass Mix)
- Cozzy D - Aphrodite (Original Mix)
- Miguel Campbell - Something Special
- Gorgon City - Real (Feat. Yasmin)
- Jessie Ware - Running (Disclosure Remix)
- Duke Dumont - The Giver
- Maceo Plex - Under The Sheets
- The Martinez Brothers - H 2 Da Izzo
- Nick Curly - Underground (Dennis Ferrer Remix)
- Eats Everything - Entrance Song
- Pirupa - Party Non Stop (Huxley Remix)
- Maya Jane Coles - What They Say

7.2 Electrohouse

- Malente, Azzido Da Bass - Hunting (Slap In The Bass Remix)
- Monkeyflip (Boys Noize Remix)
- Secret Smile - Steaming (Original Mix)
- Sick Boy - Bang Your Head (Clockwork Remix)
- Spencer and Hill - i Spy (Club Mix)
- Stefano Noferini - Move your body (Plastik Funk Remix)
- Utku S. - Robot Love (Planger & Poincue Remix)
- Zedd - Legend Of Zelda (Electrixx Remix)
- Zedd - The Legend Of Zelda (Original Mix)
- Nicole vs. TAITO feat. DGS - Crazy (TAITO Remix)
- Hirshee feat. Tonye Aganaba - So Good (Rico Tubbs Remix)
- Anton Wick feat. Evelyn Thomas - That's It (Laurent Wolf Club Mix)
- Van Date & Dirty Bass Project Ft. Mix'Usha - Step By Step (DJ Viduta Remix)
- Mattias & G80's feat. Master Freez - Get Your Hands Up (Big Room Mix)
- Ian Carey feat. Snoop Dogg & Bobby Anthony - Last Night (Dani L. Mebius Remix)
- Stream Dance Project - Thought In The Ear (Original Mix)
- Utku S. - Empty Space (Original Mix)
- Dave Darell vs Klingenberg - Die For Love (Digital Freq Vocal Remix)
- Britney Spears - I Wanna Go (Martin & Souza Club Mix)
- Akcent - Love Stoned (Eric Chase Remix)
- B.o.B feat. Hayley Williams - Airplanes (Dj PM Club Mix)
- Badboys Brothers - La Mia Canzone DAmore (Mad Players & Moe Aly Remix)
- Basic Element - I'll Never Let You know (Dj AFFecta Remix)
- Bass N Whomps Feat Disco Unit - Bass! (Original Mix)
- DJ Dee Ass - Disco Desaster (DJ Dee Ass Fall Down Remix)

- DVXL - Power (Original Mix)
- Felguk - Buzz Me (Rework 2011 Master)
- Fonzerelli feat. Digital Glitter - Feel the Love (Cut & Splice Remix)
- Ftampa - Colossus (Original Mix)
- Ivan Frost - Pump Up The Volume (Vova Baggage Remix)

7.3 Dubstep

Note that the 'album megamix' and 'dubstep megamix' were included. These mixes or mashups of tracks included on the album were not significantly longer in duration than the other tracks on the album, while (clearly) belonging to the same genre. Because of the high similarity with other tracks on the album, no skew would be expected from including them.

- Freestylers - Cracks (Flux Pavilion Remix)
- Gemini - Blue
- Jamiroquai - Blue Skies (Flux Pavilion Remix)
- Kano - Spaceship (Trolley Snatcha Remix)
- La Roux - I
- Laid Blak - Red (Chasing Shadows Remix)
- Medison - Harry feat. Skrein (Bare Noize Remix)
- Modestep - Sunlight (Official Video)
- Modestep - Sunlight
- Modestep - To The Stars (Break The Noize
- N.A.S.A. (Feat. Kanye West, Santigold
- Nero - Innocence
- Noisses - End Of
- Professor Green - Monster feat. Example (Camo
- Robyn - Call Your Girlfriend (Feed Me Remix)
- Skism - The Blank
- Skream
- Spor - Pacifica (Chasing Shadows Remix)
- Subscape - Turn Me On
- Tek-One - Broken String
- The Prodigy - Breathe (Numbernin6 Remix)
- The Streets - In The Middle (Nero Remix)
- UKF Bass Culture (Dubstep Megamix)
- UKF Dubstep 2010 (Album Megamix)
- Zeds Dead - White Satin
- Bar 9 - Piano Tune
- Barletta - Panther (Zeds Dead Remix)
- Bionic Commando (Rusko Remix)
- Black Sun Empire - Hyper Sun
- Blame ft. Camilla Marie - Star (Doctor P Remix)
- Blue Foundation - Eyes On Fire (Zeds Dead Remix)
- Booty Luv - Say It (Nero Remix)
- Boy Crisis - Dressed To Digress (Nero Remix)
- Calvin Harris - Feel So Close (Nero Remix)
- Cassius - The Sound Of Violence (Tha Trickaz Remix)
- Chapel Club - All The Eastern Girls (Flux Pavilion Remix)
- Chase
- Chasing Shadows - Amirah
- Dansette Junior - Paranoid (Official Video)
- DJ Fresh - Gold Dust (Flux Pavilion Remix)
- DJ Fresh - Louder (Doctor P

- Dubba Jonny - Home
- Emalkay - When I Look At You
- Example - Kickstarts (Bar 9 Remix) (Official Video)
- Example - Kickstarts (Bar 9 Remix)
- Example - Last Ones Standing (Doctor P Remix)
- Feed Me - Blood Red
- Feed Me - Strange Behaviour (ft. Tasha Baxter)
- Flux Pavilion - I Can
- Flux Pavilion - Lines In Wax (feat. Foreign Beggars)

7.4 Drum & Bass

- Loadstar - link to the past
- Noisia - friendly intentions
- Phace and misanthrop - desert orgy
- Wilkinson - moonwalker
- Blokhe4d - kisses and lies
- Mattix and futile - corkscrew
- Metrik - t-1000
- Icicle - dreadnaught (feat sp mc)
- Btk - things i do (spectrasoul remix)
- Tyke - the music makers (vip) (feat recipe)
- Shameboy - strobot (netsky remix)
- Modestep - feel good (the prototypes remix)
- Camo and krooked - nothing is older than yesterday
- Avicii and sebastien drums - my feelings for you (the prototypes remix)
- Danny byrd - ill behaviour (feat i-kay)
- Fred v and grafix - one of these days
- Camo and krooked - feel your pulse
- Lenzman - bittersweet part 2 (feat riya)
- Jubei - gateway
- Enei and eastcolors and noel - cracker
- Hybris - of two minds
- Rockwell - noir (ulterior motive remix)
- Jonny l - the rave
- Twisted individual - i am leg end
- Cyantific - 305
- Gridlok and prolix - tru born playa (feat mc fats
- Blokhe4d - horror show
- Chase and status - no problem
- Nero - me and you (dirtyphonics remix)
- Pendulum - crush
- Breakage - fighting fire (feat jess mills) (loadstar remix)
- Sigma and dj fresh - lassitude (sigma vip remix)
- Brookes brothers - beautiful (feat robert owens)
- Camo and krooked - mind is drifting away (feat shaz sparks) (vip mix dub)
- Danny byrd - tonight (feat netsky)
- Spy - by your side (logistics remix)
- Nu tone - shine in (feat natalie williams)
- Baby d - let me be your fantasy (j majik and wickaman remix)
- Dj hazard - busta move
- Delta heavy - space time

7.5 Hardcore

- The Dope-X & Komarovski - Milli Motherfucker (Roughsketch & Quil Remix)
- Nitrogenetics - Pledge Of Resistance (Angerfist Remix)
- Amnesys - Back 2 Zero
- Amnesys Ft. MC Axys - Destroy D Elements
- Quil - Katharsis
- Neophyte & the Viper - Peace
- N Vitral - Welcome To The Killzone
- Endymion & Nosferatu - Uphold the Future
- Ophidian & D-Passion - Breathe
- Nitrogenetics - Train of thought
- Meccano Twins - WTFisthis
- Re-Style - Wasteland (Official Bassleader 2012 Anthem)
- The Wishmaster - Pandemonium
- Art of Fighters - Toxic Hotel (Original Mix)
- Tha Playah - The Impact
- Hellsystem - Blood (Tha Playah Remix)
- Ophidian & D-Passion - Another Destiny
- Outblast - Superhero Complex
- Pandorum - Surpassed
- Mad Dog - Re-Enter The Timemachine (Amnesys Re-Enter Mix)
- Pandorum Ft. Ruffneck - Bitcore
- Masters Elite - Tied by Sound (Official Syndicate Anthem)
- The Outside Agency & Ophidian - The Infinite (Original Mix)
- Endymion - Anarchy
- Dyewitness - Masterplan (State Of Emergency & Outblast Ft. MC Syco Remix)
- Javi Boss - Faka
- D-Passion - Good Memories
- D-Passion - Dispel the Darkness
- Fracture 4 - I Never Want To See Your Face Again
- Art Of Fighters - Tears Of Blood
- Angerfist - Buckle Up & Kill
- Nosferatu Ft. Alee - Beyond Borders
- The Outside Agency - Backpack Wisdom
- Korsakoff & Outblast - Eternity (You Will Never Be Forgotten)
- Omi - Hardmony
- Anime - A-Bomb
- Miss K8 - Halucin8 (Original Mix)
- Promo Feat. Tha Playah & Snowflake - Open
- Ak Industry - Reloaded (Feat. Igneon System & N-Vitral)
- Tommyknocker - T-2012
- Nitrogenetics - Mu-Sick (Hellsystem Remix)
- AniMe - Be a god (Endymion remix)
- Paul Elstak & Partyraiser - Back from the Dead
- The Outside Agency - Ghetto Blast
- Promo Feat. D-Passion - Analog
- Kartel - Simfony
- Dyprax & Predator - Blood Cycle RedMusic.pl
- Evil Activities & Panic feat. Mc Alee - Never Fall Asleep (Tha Playah Remix)
- Negative A & Counterfeit - Beast Wars
- Hellsystem - Salvation

7.6 Hardstyle

- Tuneboy feat. Soleo - Six
- Titan - Hooked
- Alpha$\hat{2}$ & E-Force - Boogeyman
- The Machine - Rollin
- Headhunterz Feat. Krewella - United Kids Of The World
- Outlander - Telepathy
- The Pitcher - Back To Basics
- D-Block & S-te-Fan - Next Level
- Atmozfears - Destroy
- Thyron - Hell's Fire
- Linkin Park & Steve Aoki - A Light That Never Comes (Coone Remix)
- Armin van Buuren - Shivers (Frontliner Remix)
- Atmozfears - Connected
- Gunz For Hire - Swagger
- Brennan Heart and Zatox - Fight The Resistance
- Code Black - Pandora
- Frequencerz & B-Front - Fatality
- Audiotricz - Infinite
- Rebourne - Trigger
- Tatanka - Shine Again
- Noisecontrollers - What
- Wildstylez feat. Cimo Frankel - Back To History (Intents Theme 2013)
- The R3belz - Earth
- Noisecontrollers - All Around The World
- Atmozfears - F#CKING J#MP
- Hard Driver - Exploration (Hard Bass 2014 Anthem)
- Darren Styles and Gammer - You & I (Da Tweekaz Remix)
- D-Block & S-Te-Fan - Built This City
- Frontliner feat. Nikkita - Death Of A Demon (B-Front Remix)
- Da Tweekaz feat. Anklebreaker - Music Is My Drug
- Hard Driver - The Red Kill
- Rebourne - Horizon
- Svenson & Gielen - Twisted (Sound Freakerz Remix)
- Omegatypez - New Moon
- Audiofreq - Guardians Of Time (Reverze 2014 Anthem)
- Technoboy - Wargames
- Thyron - Reincarnation
- Waverider - For The Music
- Max Enforcer - Lost In Paradise
- Hard Driver - Nature Of Blue

7.7 Trance

- Game Over (Radio Edit) - Heatbeat
- The Future (Radio Edit) - Marlo
- Six Zero Zero (Radio Edit) - Jorn Van Deynhoven
- Drop (Radio Edit) - Shogun
- The Light (Radio Edit) - Omnia
- Bang! (Radio Edit) - Alex M.O.R.P.H. & Jerome Isma-Ae
- Gunsmoke (Radio Edit) - Bjorn Akesson
- Seize the Day (Radio Edit) - Ralphie B & Mesh
- Aureolo (Radio Edit) - Rex Mundi
- Moscow Subway (Radio Edit) - Alexander Popov

- Sukha (Radio Edit) - Toby Hedges
- The Spiritual Gateway - Transmission Theme 2013 (Radio Edit) - Markus Schulz
- Beyond the Time (Radio Edit) - A.R.D.I.
- Sacramentum (Andrew Rayel Aether Radio Edit) - Bobina & Andrew Rayel
- Raptor (Radio Edit) - Digital X
- Earthbeat (Radio Edit) - Giuseppe Ottaviani
- Gladius (Radio Edit) - Arisen Flame
- Darksiders (Radio Edit) - Chris Schweizer & Tomas Heredia
- Uppercut (Radio Edit) - Airbase
- Farewell To the Moon (Alexander Popov Radio Edit) - York
- Seven Cities (Thomas Datt Radio Edit) - Solarstone
- Crash (Radio Edit) - Ram
- Exodus (feat. D-Sharp) [Radio Edit] - Driftmoon & Andy Blueman
- Superstar (Radio Edit) - Luke Bond
- Caesar (Radio Edit) - Ayda
- Whiteout (Radio Edit) - WaCh & Leven Mervox
- Requiem (Radio Edit) - Steve Haines
- Alonism (Radio Edit) - Eco
- Elusive (James Dymond Radio Edit) - Paul Trainer
- Apache (Official Radio Edit) - Fisherman & Hawkins
- Violetta (Radio Edit) - Orjan Nilsen
- The Expedition - A State of Trance 600 Anthem (Radio Edit) - Armin van Buuren & Markus Schulz
- Character (Radio Edit) - Mark Sixma
- Skylarking (Ilan Bluestone Radio Edit) - BT
- Under the Gun (feat. Leigh Nash) [Rank 1 Radio Edit] - Conjure One
- Jar of Hearts (feat. Christina Novelli) [Official Radio Edit] - Dash Berlin
- Bulldozer (Radio Edit) - David Gravell
- Laguna (Radio Edit) - Protoculture
- I Be (Radio Edit) - Andy Moor
- The Evil ID (Radio Edit) - Max Graham

7.8 Rock

- Patti Smith Group - Because The Night
- Santana - She's Not There
- Him - Solitary Man
- Toto - Hold The Line
- Blue yter Cult - (Don't Fear) The Reaper
- Gillan - New Orleans
- Fleetwood Mac - Black Magic Woman
- Stray Cats - Runaway Boys
- Ian Hunter - Once Bitten Twice Shy
- The Edgar Winter Group - Frankenstein
- Matt The Hoople - Roll Away The Stone
- Rick Springfield - Jessie's Girl
- Cheap Trick - I Want You To Want Me
- Redbone - The Witchqueen Of New Orleans
- Adam Ant - Vive Le Rock
- Living Colours - Love Rears It's Ugly Head
- Boz Scraggs - Lido Shuffle
- Gillan - Trouble
- John Farnham - You're The Voice
- Kenny Loggins - Danger Zone

- Argent - Hold Your Head Up
- Soul Alysum - Runaway Train
- Europe - The Final Countdown
- Reef - Place Your Hands
- Alice Cooper - Poison
- Kula Shaker - Hush
- Run-DMC With Aerosmith - Walk This Way
- Suede - Filmstar
- Meat Loaf - Bat Out Of Hell
- Spin Doctors - Two Princes
- Sweet - Hell Raiser
- The Calling - Our Lives
- Bowling For Soup - Girl All The Bad Guys Want
- Love-Hate - Wasted In America
- Hundred Reasons - If I Could
- Lordi - Hard Rock Hallelujah
- Belinda Carlisle - Live Your Life Be Free
- Ram Jam - Black Betty
- Sophie B. Hawkins - Damn, I Wish I Was Your Lover
- The Stranglers - All Day & All Of The Night

7.9 Jazz

- Jimmy Rushing - Pennies from Heaven
- Benny Goodman - Sing Sing Sing
- Ed Bentley - Black Coffee
- Mel Torme - Lullaby of Birdland
- Ronnie Scott - A Night In Tunisia
- Frankie Laine - That's My Desire
- Thelonious Monk - Off Minor
- Ahmad Jamal - Patterns
- Stan Getz - I've Got You Under My Skin
- Erroll Garner - The Lady Is a Tramp
- Sarah Vaughan - Embraceable You
- Art Blakey - Moon River
- Eddie Lockjaw Davis - Bewitched,Bothered and Bewildered
- Lena Horne - Old Devil Moon
- Benny Carter - Crazy Rhythm
- David Benoit - Here There And Everywhere
- Ray Anthony & His Orchestra - Dragnet
- Lawrence Welk - Smoke Gets In Your Eyes
- Gene Krupa - Drummin' Man
- June Christy - Lullaby In Rhythm
- Ted Heath - East of the Sun
- Al Hirt - Tuxedo Junction
- Artie Shaw - Begin the Beguine
- Count Basie - Jumpin' at the Woodside
- Louis Prima - Buona Sera
- Woody Herman - The Good Earth
- Django Reinhardt - Between the Devil and the Deep Blue Sea
- Nat King Cole - Route 66
- Frankie Laine - Dream a Little Dream of Me
- Scott Joplin - The Entertainer
- Eartha Kitt - Let's Do It, Let's Fall In Love

- Count Basie - Honeysuckle Rose
- Marilyn Monroe - I Wanna Be Loved By You
- Jo Jones - Satin Doll
- Julie London - Cry Me a River
- Mel Powell - S Wonderful
- Henry Mancini - A Quiet Gass
- Lionel Hampton - Lullaby of Birdland
- Gene Krupa - Let Me Off Uptown
- Mildred Bailey - My Melancholy Baby

8 Appendix B: Code

8.1 processfolder.m

input: folder containing mp3-files

```
function processfolder = processfolder(dirname)

addpath(genpath('./MIRtoolbox/'))

dirname2 = strcat(dirname,'*.mp3')

files = dir(dirname2)

for file = files'
    processmp3(strcat(dirname,file.name), file.name);
end
```

8.2 processmp3.m

```
function processmp3 = processmp3(file, output)

addpath(genpath('./MIRtoolbox/'))

size = mp3read(file, 'size')
mp3 = mp3read(file, size(1)/4, 1, 4);
s1 = miraudio(mp3)
s1_rms = mirrms(s1);
s1_len = mirlowenergy(s1);
s1_env = mirenvelope(s1);
s1_set = mironsets(s1);
s1_att = mirattacktime(s1);
s1_ats = mirattackslope(s1);
s1_atl = mirattackleap(s1);
s1_dens = mireventdensity(s1);
s1_temp = mirtempo(s1);
s1_flux = mirfluctuation(s1);
s1_pulse = mirpulseclarity(s1);
s1_bri = mirbrightness(s1);
s1_roll = mirrolloff(s1);
s1_mfcc = mirmfcc(s1);
s1_inhar = mirinharmonicity(s1);
s1_rough = mirroughness(s1);
s1_reg = mirregularity(s1);
s1_pitch = mirpitch(s1);
s1_reg = mircepstrum(s1);
s1_chroma = mirchromagram(s1);
s1_keyst = mirkeystrength(s1);
s1_key = mirkey(s1);
s1_keys = mirkeysom(s1);
s1_mode = mirmode(s1);
s1_tone = mirtonalcentroid(s1);
s1_harm = mirhcdf(s1);
```

```
mirexport(strcat('features_',output,'.dat'), s1, s1_rms, s1_len, s1_env, s1_set, s1_att, s1_ats,
s1_atl, s1_dens, s1_temp, s1_flux, s1_pulse, s1_bri, s1_roll, s1_mfcc, s1_inhar, s1_rough,
s1_reg, s1_pitch, s1_reg, s1_chroma, s1_keyst, s1_key, s1_keys, s1_mode, s1_tone, s1_harm)
```

8.3 automaticParameterSelection.m

```
function [bestc, bestg, bestcv] = automaticParameterSelection(trainLabel, trainData, Ncv, option)
[N, D] = size(trainData);

if nargin>3
    stepSize = option.stepSize;
    bestLog2c = log2(option.c);
    bestLog2g = log2(option.gamma);
    epsilon = option.epsilon;
    Nlimit = option.Nlimit;
    svmCmd = option.svmCmd;
else
    stepSize = 5;
    bestLog2c = 0;
    bestLog2g = log2(1/D);
    epsilon = 0.005;
    Ncv = 3; % Ncv-fold cross validation cross validation
    Nlimit = 100;
    svmCmd = '';
end

% initialise some auxiliary variables
bestcv = 0;
deltacv = 10^6;
cnt = 1;
breakLoop = 0;

while abs(deltacv) > epsilon && cnt < Nlimit
    bestcv_prev = bestcv;
    prevStepSize = stepSize;
    stepSize = prevStepSize/2;
    log2c_list = bestLog2c-prevStepSize: stepSize: bestLog2c+prevStepSize;
    log2g_list = bestLog2g-prevStepSize: stepSize: bestLog2g+prevStepSize;

    numLog2c = length(log2c_list);
    numLog2g = length(log2g_list);

    for i = 1:numLog2c
        log2c = log2c_list(i);
        for j = 1:numLog2g
            log2g = log2g_list(j);
            % With some precal kernel
            cmd = ['-c ', num2str(2^log2c), ' -g ', num2str(2^log2g),' ',svmCmd];
            cv = get_cv_ac(trainLabel, trainData, cmd, Ncv);
            if (cv >= bestcv),
                bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
                bestc = 2^bestLog2c; bestg = 2^bestLog2g;
            end
            disp(['So far, cnt=',num2str(cnt),' the best parameters, yielding Accuracy=
```

```
            ',num2str(bestcv*100),'%, are: C=',num2str(bestc),', gamma=',num2str(bestg)]);
            % Break out of the loop when the cnt is up to the condition
            if cnt >= Nlimit, breakLoop = 1; break; end
            cnt = cnt + 1;
        end
        if breakLoop == 1, break; end
    end
    if breakLoop == 1, break; end
    deltacv = bestcv - bestcv_prev;

end
disp(['The best parameters, yielding Accuracy=',num2str(bestcv*100),'%, are:
C=',num2str(bestc),', gamma=',num2str(bestg)]);
```

8.4 classifyUsingCrossValidation.m

This script is a highly adapted/modified version of a script originally found at [10].

```
function [predictedLabel, decisValueWinner, totalAccuracy, confusionMatrix, order] =
classifyUsingCrossValidation(label, data, run, Ncv_classif, option)

if exist('option','var')
    c = option.c;
    g = option.gamma;
    genres = option.NClass;
    svmCmd = option.svmCmd;
end

% Prepare/initialize some matrices to store some information
order = zeros(genres,Ncv_classif);
totalAccuracy = zeros(1,Ncv_classif);
predictedLabel = label*0;
decisValueWinner = label*0;
dec_values_concat = zeros(0,9);
label_out_concat = zeros(0,9);
acc_global = 0;

% SVM parameters
% parameters are obtained from cross validation process
cmd = ['-c ',num2str(c),' -g ',num2str(g),' ','-b 1',svmCmd];

for ncv = 1:Ncv_classif

    % Pick one fold at a time
    testIndex = run == ncv;
    trainIndex = run ~= ncv;
    trainData = data(trainIndex,:);
    trainLabel = label(trainIndex,:);
    testData = data(testIndex,:);
    testLabel = label(testIndex,:);
    NTest = sum(testIndex,1);

    % Train the SVM in one-vs-one mode
    model = svmtrain(trainLabel, trainData, cmd);
```

```
    % Classify samples using 1v1 model
    [predict_label, accuracy, dec_values] = svmpredict(testLabel, testData, model,'-b 1');
    [decis_value_winner, label_out] = max(dec_values,[],2);
    predictedLabel(testIndex) = label_out;
    decisValueWinner(testIndex) = decis_value_winner;

    % Concatenate output.
    dec_values_concat = [dec_values_concat; dec_values];
    label_out_concat = [label_out_concat; label_out];
    acc_global = (accuracy + acc_global)/2;
end

% write variables to workspace
assignin('base','dec_values',dec_values_concat);
assignin('base','predict_label',label_out_concat);
assignin('base','model',model);
assignin('base','global_acc',acc_global);
```

8.5 OptionCV

```
optionCV.stepSize = 5;
optionCV.c = 1;
optionCV.gamma = 1/numFeatures;
optionCV.stepSize = 5;
optionCV.bestLog2c = 0;
optionCV.bestLog2g = log2(1/numFeatures);
optionCV.epsilon = 0.005;
optionCV.Nlimit = 100;
optionCV.svmCmd = '-q';
optionCV.NClass = 9;
```

8.6 svmprep.m

```
genres = grp2idx(datafile.Genre); % create numerical genre labels
genreList = unique(genres); % create a list of unique genres
opts = '-s 0 -t 2 -c 1 -b 1'; % -b 1 allow probability functionality via
libsvm # 'default' options, we'll select better options using CV.
numFeatures = size(datafile,2)-1; % calculate the number of features
zdata = zeros(size(datafile,1),1); % initialise zscore-list

% normalise data
for i = 2:numFeatures
zdata = datafile(:,i);
zdata = zscore(double(zdata));
datafile(:,i) = dataset(zdata);
end

datafile(:,1) = dataset(genres); % prepare dataset in double format
datafileD = double(datafile); % copy dataset in double format, for libsvm
datafileLabels = datafileD(:,1);
```

8.7 svmprep2.m

```
% revert transposing in previous step, since libsvm uses rows
valSet = valSet';
testSet = testSet';

% strip labels to prevent classification bugs
valSetLabels = valSet(:,1);
testSetLabels = testSet(:,1);

valSet(:,1) = []; % discard labels
testSet(:,1) = [];
```

9 Appendix C: Dropped features

- AttackTime_PeriodFreq
- AttackTime_PeriodAmp
- AttackSlope_PeriodFreq
- AttackSlope_PeriodAmp
- AttackLeap_PeriodFreq
- AttackLeap_PeriodAmp
- HarmonicChangeDetectionF5
- HarmonicChangeDetectionF
- HarmonicChangeDetectionF1
- HarmonicChangeDetectionF2
- HarmonicChangeDetectionF3
- HarmonicChangeDetectionF4
- Fluctuation_Mean