KTH Computer Science
and Communication

# Machine Learning Applied to Playlist Generation

Duis autem vel eum iruire dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat null

ERIK AALTO

Master's Thesis at Spotify and CSC
KTH Supervisor: Carl Henrik Ek
Company Supervisor: Boxun Zhang
KTH Examiner: Danica Kragic

TRITA xxx yyyy-nn

# Abstract

This is a skeleton for KTH theses. More documentation regarding the KTH thesis class file can be found in the package documentation.

# Referat

## Lorem ipsum dolor sit amet, sed diam nonummy nibh eui mod tincidunt ut laoreet dol

Denna fil ger ett avhandlingsskelett. Mer information om LaTeX-mallen finns i dokumentationen till paketet.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Mauris purus. Fusce tempor. Nulla facilisi. Sed at turpis. Phasellus eu ipsum. Nam porttitor laoreet nulla. Phasellus massa massa, auctor rutrum, vehicula ut, porttitor a, massa. Pellentesque fringilla. Duis nibh risus, venenatis ac, tempor sed, vestibulum at, tellus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos.

# Contents

# Chapter 1

# Introduction

### 1.0.1 What are Recommender Systems?

Recommender systems provide an automated way to filter information of interest for a certain user, possibly also taking time into account. A famous example of recommender systems is the product recommendation once initiated at Amazon, "Users who bought this product also bought". Another example of recommender systems are the movie recommendations provided by Netflix. Movie recommendations are interesting and non-trivial as a specific user at a certain time is likely to not be interested in the majority of movies provided by Netflix. A last example of recommendation could be restaurant recommendation, where time and context are important factors. Recommending a simple hamburger restaurant is not likely to be of interest at date night, but it might be the perfect recommendation while driving the kids home after the saturday morning soccer game.

### 1.0.2 What is Spotify?

Spotify is a music streaming service which charges premium users a fee and presents free users with ads. Record companies are then paid according to the popularity of the tracks for which they hold digital rights. Spotify was launched in October 2008 and today has over 60 million active users, from which over 15 million are paying for the premium user service.

### 1.0.3 Problem Background

Recommendation in general and music recommendation in specific are areas of current interest. Systems for automatic music recommendation already exist, such as the Spotify Radio and the Spotify recommendations of songs and artists. Examples of other actors within the music industry who are using music recommendation are Pandora, last.fm and iTunes.

One type of music recommendation is item recommendation. Given that a user has a preference for a specific item, another supposedly similar item can be recommended. Item recommendation can consist of recommending songs, artists or albums for example.

Another type of music recommendation is to provide curated playlists, playlists that have been made by a domain expert. These playlists aim to have an as good as possible flow of songs. These playlists exist within the Spotify client. Curated playlists can be made to fit a certain genre, such as a rock playlist or a reggaeton playlist, they might try to fit a particular mood such as party, or a playlist might be fit to a particular activity such as workout. To not confuse the reader, all the previous contexts for which a particular playlist might be fit will be referred to as contexts or playlist contexts.

This master thesis seeks to be a step on the way of automatically generating playlists, given a predefined playlist context and thus extend the current area of music recommendation.d

### 1.0.4 Project Aim

The work of this thesis aims to create a method for generating playlists that contain a good sequence of songs, in a qualitative sense, by the use machine learning given a playlist context.

### 1.0.5 Research Questions

- How can candidate songs for playlist generation, given a playlist context, be ranked through the use of machine learning techniques?

- How can a representation of the order of songs in a generated playlist be created?

### 1.0.6 Previous Work

Previous work within the recommender system domain mainly focuses on either collaborative filtering, content based approaches or a hybrid of these two approaches. Both of these approaches try to infer a user ranking for a specific item. An item would in the context of music recommendation be a song, artist or album.

### 1.0.7 Collaborative Filtering

Collaborative filtering focuses on userÂ´s past behaviour. From this past behaviour of a specific user and past behaviour of similar users the ranking for the specific user for a certain item is inferred. Collaborative filtering suffers from something called the cold start problem, which occurs when the ranking for a specific item and user is inferred when there is no track of the current user behaviour. An example of collaborative filtering applied to music recommendation is the recommender system used by last.fm.

### 1.0.8 Content Based Approaches

Content based approaches look at discrete features of items and tries to infer a similarity between two items given their similarity of features. An example of a content based approach within music recommendation is the recommendations made by Pandora.

### 1.0.9 Related Work

Earlier attempts of playlist generation has been made by Microsoft Research. Ragno, Burges and Herley has made a model for playlist generation that can take any type of ordered playlist material, such as curated playlists or albums, as training data, and constructs an undirected graph between songs that are within the reach of a nth-order Markov model. In this graph nodes constitute songs and edges get their weights depending on how many times two songs fulfil the nth-order Markov property. Once this is done the undirected graph is converted into a directed graph where edges weightÂ´s, the transition probabilities, are normalized by the sum of outgoing weights from each node. Once the undirected graph is made a playlist can be generated by selecting an initial seed song and simply performing a random walk in the graph. This model assumes that the connectivity between songs does not have to take order into account and that one can prevent playlist drifting by adding higher order Markov properties.

From a contextual playlist generation perspective a problem with the approach taken by Rango, Burges and Herley is that if you generate a playlist from a random walk you cannot chose the playlist context for the generated playlist on before hand. Another problem with the probabilistic graphical model approach to playlist generation is that the graph created during training phase only works for songs that are in the training data set. This model is not generalizable so you cannot get a similar playlist to a playlist you like, but with different songs.

Another approach to playlist generation is to use gaussian processes, this approach has been taken by Platt et al, also at Microsoft Research. Here the authors try to learn a gaussian process prior from training data. This prior is then used together with a set of songs, for which a user has expressed preference, to generate a playlist given an initial seed song. In the training phase a blend of linear kernels is used to learn the relationship of meta data features among songs that come in sequence. The coefficients for each linear kernel is learnt by finding the coefficient that minimizes the difference between the empirical covariance between songs and the value given by the linear kernel. Empirical covariance is in this case as simple as whether the training data songs belong to the same album or not. Once the training phase is done the playlist generation phase consists of predicting the user preference for each song in a set of candidate songs, i.e. the f-star function in this case is the predicted user preference for a song. The f-star value is calculated by weighing the blend of linear kernels between a seed song and each candidate song with a factor. This factor is the sum of similarity between the initial seed song and each user preference song weighted by how central each user preference song is in the preference space. Playlist generation is then done by simply choosing the songs with highest f-star value.

This model generalizes to new songs, but the user preference space is seen as one single space. This is a simplification of reality where a user preference space is probably divided into several categories, for example a workout preference space and a chill-out preference space, something the model provided does not take into account, which can be claimed as a weakness in terms of playlist context generation. Neither does the model take the ordering of songs into account.

# Chapter 2

# Methodology

### 2.0.10 Assumptions

The first assumption for this thesis is that curated playlists suited for a specific context are suitable training data to create a model that generates playlists suited to the same playlist context.

The second assumption made is that features that belong to each track in a curated playlist contain enough information to create a representation of the context this curated playlist is made for.

### 2.0.11 Data

From the Spotify hadoop cluster all available playlists where extracted and then filtered based upon whether they were created by Spotify playlist curators or not. Once playlists were filtered, feature data consisting of discrete values for genre, mood and tempo were added to each track within the selected subset of playlists.

### 2.0.12 Data Preprocessing

Feature data was encoded in JSON format and extracting this data proved difficult with the standard R JSON packages why Python was used to preprocess data into csv format.

### 2.0.13 Exploratory Data Analysis

To get an overview of whether features of tracks in a curated playlist relate to each other within the playlist correlation plots were made. The idea behind plotting correlations instead of covariances is that the magnitude of the correlation shows the strength of the linear relationship between features, while a covariance plot would be polluted should different features be on different ranges. By plotting correlations the problem of calculating correlations for features with zero variance, given a playlist context, emerged. This problem was solved by setting the correlation for feature relations with zero covariance to zero. It can be argued whether this is mathematically correct or not. But the approach can be motivated by the fact that plots are done to get an intuition of the data and a correlation of zero for features with zero covariance gives a better intuition of relationships in the data set compared to setting the correlation to one.

### 2.0.14 Playlist Characteristics

Explaining the characteristics for a certain playlist context could be seen as equivalent of explaining the variance of features for tracks, given a curated playlist suited to the specific playlist context. Therefore extracting the main characteristics for a playlist context can be done by extracting the principal components, for the curated playlist representing that playlist context, up to a certain threshold for the variance explained. Using this approach extracting eigenvectors for the covariance matrix, rather than correlation matrix, is a motivated choice. The motivation behind this choice is that scaling the covariance matrix to a correlation matrix is a nonlinear transformation. If we want to use apply the principal components of a correlation matrix to the original data, then the original data need to undergo the same transform as transforming covariances to correlations. For a data set where each curated playlist makes up less than one percent of the total data it would be impractical to transform the original data over and over as we extract the principal components for each playlist context. Using the covariance matrix for extraction of features is therefore motivated as the principal components of the covariance matrix can be directly related to the existing data.

### 2.0.15 Handling zero variance terms

Even though there are no zero variance terms in the whole data set, there are some terms that have zero variance within a certain curated playlists. These terms will not be handled by the principal components describing a playlist context, as principal components describe the variance of a playlist. Despite not being handled by the principal components zero variance terms might still have an important role in describing a playlist context. For example, if we have a curated jazz playlist then it is probably an important factor that all of the tracks in this playlist have a zero value for rap (or imagine the opposite, if they have a constant non-zero value for rap).

### 2.0.16 Selecting candidate songs for a playlist context

The process of selecting candidate songs given a specific playlist context is an interesting and ambiguous problem without a given approach. Earlier work is focused mainly on item to item recommendation, i.e. recommending similar items of the same type given preferences for items of a certain times. But when it comes to selecting appropriate songs for a playlist context the items are of different kinds. The goal is to recommend songs, one type of item, given a playlist describing a playlist context, which is another type of item.

One initial idea to select songs for a given playlist context could be that songs are either good candidates or not. This is a reasonable assumption, as for example for a rock classics playlist context then songs are either rock classics or not. Given that this is a binary classification problem, an efficient two class classifier might seem as a good idea at a glance. A support vector machine, SVM, is an optimal two class classifier by definition, as a SVM maximizes the margin between classes, and has the capability of multi class classification with the one versus all approach. There is however one problem with support vector machines, or any classifier that requires training data within contextual playlist generation. The problem is that it is easy to define training data which labels a song as belonging or not belonging to a certain playlist. But it is hard to define what songs that belong to other playlists, than the one describing a specific playlist context, which are still relevant for that playlist context. For example a song belonging to a House Workout playlist may very well be a suitable candidate for a House Party playlist. It is actually often the case that many songs belong to several playlists, describing different playlist contexts. Given this example a discriminative model turns out to be a bad fit for the problem this thesis is trying to solve. If a song belongs to a House Party playlist then it is reasonable to assume that it would be outside the margin defining a House Workout playlist if feeded to a SVM, even though this particular song might very well be a suitable match for the House Workout playlist. This rules out the use of SVMs for the purpose of this thesis, as SVMs need to know the mapping between songs and playlist contexts to work. The same mapping that we are trying to find.

A second idea to selecting songs suitable for a specific playlist context would be to use centroid based clustering. The wikipedia definition of clustering is as follows: "clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters)". One could for example cluster all tracks in curated playlists and then simply assign each song that is not part of a curated playlist to the cluster providing the best fit for each track. But one problem is that there is not a one to one mapping between tracks and playlists, one playlist can contain many tracks and one track can belong to many playlists. This is different from clustering where each cluster consists of many points, but each point only belongs to one cluster, which yields centroid based clustering impropriate for the scope of this thesis.

A third approach to finding candidate songs given a playlist context would be to tweak the normal usage of collaborative filtering. The common approach of collaborative filtering is to use a sparse matrix to infer the rating of items for one user given the ratings of similar users. What can be done instead is to use binary ratings and instead of inferring ratings for a user one could infer ratings for songs given a playlist. What this means is that playlists that contain the same songs as a playlist describing a playlist context one is interested in will be used to infer songs that are good matches for the specified playlist context.

A last approach for track candidate selection would be to use the subspace method. Given that the principal components for a playlist, describing the variance of that playlist suited for a playlist context, are at hand, one can simply treat each track as a vector rather than a point. Each vector can then be projected into the principal component space for that playlist context. The underlying assumption is then that points that have a low relative change in magnitude under projection are well described by the characteristics defining the playlist context, and thus good candidates. Tracks that are not well described by the playlist context characteristics on the other hand, will change under projection and will therefore also have a high relative change in magnitude. There are however problems with this approach. Lets say that we have a playlist context that is defined by variance in the dimensions Jazz, Blues and Rap and our vector space consists of the dimensions Jazz, Blues, Rap and Rock. If we then have a song that is characterized by Jazz and Blues only, then this song will go unchanged under projection. As the relative change in magnitude is none then this song will be suggested as a suitable candidate for the Jazz, Blues, Rap playlist context. However a playlist context consisting of Jazz, Blues and Rap is likely to be pretty peculiar and a song characterized by Jazz and Blues only is not likely to be a suitable match for such a playlist. Another problem would be songs that consists of zero values for all features, these songs would also go unchanged under any playlist context projection, but are not likely to be good candidates for all playlist contexts. Further, the subspace method is a linear transformation and it can be questioned if a linear transformation is powerful enough to describe the necessary mappings.

### 2.0.17 First Track Manifold

After studying linear relationships among features within playlists the study was extended to see if there were linear relationships among the ordering of songs as well. To do this the subspace method was used again. But instead of extracting the principal components for the variance of a playlist, the principal components describing the variance of all the start songs of all curated playlists were extracted. Once this was made a sample of songs from the curated playlist data set were projected into the first track manifold space created by the principal components of the first tracks. This was made to see if start tracks would have higher ranking, i.e. lower relative change in magnitude, than other tracks.

### 2.0.18 Playlist Comparison

As principal components were chosen to describe playlists, it is reasonable to assume that if principal components analysis works well for describing playlist characteristics, then the same approach should also work well for comparing playlists. Playlists were compared pairwise. To compare two playlists all eigenvectors from each playlist were multiplied by each other. By doing this the cosine measure of vector similarity for each pair of vectors was obtained. The problem with this approach is that it gives an unbalanced comparison. By simply looking at the similarity of eigenvectors implies that eigenvectors corresponding to low eigenvalues have the same importance as eigenvectors corresponding to high eigenvalues. For the reader uninitiated with eigenvalues this means that components explaining a high part of the characteristics of a playlist are regarded an equal importance as components explaining a low part of playlist characteristics. To remedy this problem the cosine score between eigenvectors from each playlist was scaled by the square root of the product of the corresponding eigenvalues. The result obtained from this multiplication was a matrix. To rank the similarity between these matrices some type of transformation from a matrix to a single value is needed. Several approaches were tested and are covered under the evaluation section.

# Appendix A

# RDF

And here is a figure

**Figure A.1.** Several statements describing the same resource.

that we refer to here: A.1