Automatic Playlist Generation

XINGTING GONG AND XU CHEN

Stanford University gongx@stanford.edu xchen91@stanford.edu

I. Introduction

Digital music applications have become an increasingly popular means of listening to music. Applications such as Spotify allows the user to add songs to his/her playlist without downloading the song to his/her computer. The user can also be recommended songs by Spotify through Spotify's "Discover" option. Pandora—an online radio—generates a radio station based on a single user-inputted artist, genre, or composer. For these types of applications, applying algorithms to learn user preferences is extremely important. In this report, we explore two different methods to generate automatic playlists based on user-input:

1. Gaussian Process Regression (GPR):

This method takes in a set of seed songs (which can contain as little as a single song) inputted by the user to train a preference function that predicts the user preference for a new song to be considered for the playlist.

2. SVM + HMM:

For this method, we assume that the user has generated a large number of seed songs (i.e. a user has liked hundreds of songs are pandora throughout the course of a year). For this method we also require a set of low-preference songs (i.e. the user has skipped hundreds of songs on pandora over a year). With a large training set of labelled data, we can apply classification algorithms such as SVM to determine if a new song will be liked or disliked by the user.

Because we believe timbre to be an important predictor for music, we combine the SVM with an HMM to model the timbre spectra.

These methods are described in much greater detail in section IV.

II. Related Work

Automatic Playlist Generation can be a difficult task due to the fact that the user will often provide only a few seed songs. With such a small training set, it can be difficult to train a sensible similarity metric. A paper by Zheng et. al. at Microsoft Corporation devised a novel "Kernel Meta-Training" (KMT) method to

mitigate the problem of a small training set. Instead of designing a machine learning problem to train only on the user-provided seeds, the Microsoft group gathered 174,577 songs from 14,198 albums as a kernel "meta"training set. The idea is that songs placed on the same album are similar, and so it is appropriate to train the similarity metric on the set of 174,577 songs. However, their selection of musical features were mostly qualitative, and consisted of: genre (i.e. jazz, rap), subgenera (i.e. heavy metal), mood (i.e. angry, happy), style (i.e. East coast rap, Gangsta rap), rhythm type (i.e swing, disco), rhythmic description (i.e. funky, lazy), and vocal code (i.e. duet, instrumental). We felt that some of these features were not very well defined and seemed redundant in the musical qualities they were trying to capture. The example features classified under genre, subgenera, and style, for example, seem extremely interchangeable amongst the 3 categories. For our first method using GPR, we aim to expand upon the Microsoft group's work by applying KMT to data from the Million Song Dataset (described in section III). Briefly, the Million Song Dataset provides quantitative features such as tempo in beats-per-minute, or loudness in decibels, which we will instead use as features to train the similarity metric.

Our second method falls into the more standard category of binary classification problems. More notably, we want to expand upon a standard SVM by modeling the timbre sequence as a HMM. There has been a lot of research on using timbre to analyze complex instrumental textures and even rhythmic styles [2-4]. Due to the importance of timbre in characterizing sound, we believe that an SVM armed with HMM can be an effective classifier on large training sets.

III. Dataset and Features

We obtained our data from the Million Song Dataset, a dataset compiled by Columbia University's Laboratory for the Recognition and Organization of Speech and Audio and The Echo Nest. The entirety of this dataset consists of audio features and metadata for a million popular songs. For the sake of practicality, we down-

loaded the 10,000 song subset.

The dataset contains approximately 54 features per track. Of this set we hand selected a subset of 5 features upon which to perform our analysis:

- 1. Genre: Each track can consist of anywhere from 0 to multiple user-supplied genre tags from the MusicBrainz website. To keep our analysis simple, we randomly assigned each track to one its genre tags. Each track is therefore labelled by an integer that corresponds to a particular genre.
- 2. Tempo: The estimated tempo in BMP. To discretize this feature, we binned the tempo values by increments of 20.
- 3. Loudness: The average loudness of the song in dB. We binned these values by increments of 5 dB.
- 4. Decade: We included the decade in which the song was released as a feature. The motivation behind this is that songs produced in the same decade sound similar in style.
- 5. Timbre: Timbre is represented as a $12 \times N$ matrix of Mel-frequency cepstral coefficients, where N is the number of segments. Each column of the matrix is thus a 12 dimensional vector representing the 12 MFCCs of a particular time segment. We processed timbre differently for each of our two methods:

GPR: For this method, we limited ourselves to tracks with $N \ge 200$ and randomly selected 200 rows of the timbre matrix for these tracks.

SVM + HMM: Since each column of the timbre matrix is obtained from a different time segment, it makes more sense to represent the timbre matrix as a hidden markov model. We trained an HMM for each of the two song sets representing songs "liked" and "disliked" by the user (i.e. added or not added to the user's Spotify playlist from the application's list of recommendations). The loglikelihoods of each track given each of the HMM models are then used as features for the SVM.

Table 1: Feature Vector Examples

Features	Example raw values	Discretized/Processed values
Genre	rock, indie, pop, hip-hop	Integer values 0 to 9
	country, jazz, metal, folk	
	rap, dance	
Tempo	130.861, 122.174, 80.149	6, 6, 4
Year	1989, 1999, 2007	198, 199, 200
Avg. Loudness	-13.366, -7.928, -15.367	-2, -1, -3
Timbre (GPR)	12 ×N matrix of MFCCs	randomly selected 200 rows
Timbre (SVM+HM)	$12 \times N$ matrix of MFCCs	loglikelihoods under each HMM

IV. Methods

Gaussian Process Regression: The first of our main methods makes use of Gaussian Process Regression. In a Gaussian process (GP), any finite subset of the points in our domain space satisfies a multivariate Gaussian distribution. For automatic playlist generation, our domain space consists of the possible user preferences f for some song which we wish to predict. To simplify calculations, the mean of the GP is often assumed to be 0, which makes sense in our case since it is reasonable to assume that in the space of all songs, a user will probably not want to listen to most of them.

Let $seedSongs = \{x_i\}_{i=1}^N$ be the set of user-inputted songs that serve as the "seed" for which we will generate a playlist around, where x_i denotes the feature vector for seed song i. Let f_i denote the true user preference for these songs (though f_i can in principle take on any real value, for simplicity we assume it is approximately 1 with noise σ if the user selects the song as a seed). Let f_* denote the user preference for some song x_* that we want to predict. Then the joint distribution $[f_i, f_*]$ is Gaussian:

$$\begin{bmatrix} f_i \\ f_* \end{bmatrix} = \mathcal{N}(0, K) \tag{1}$$

where *K* is the covariance matrix.

Since $[f_i, f_*]$ is jointly Gaussian, the conditional distribution $P(f_*|f_i)$ is therefore also a Gaussian with parameters:

$$P(f_*|f_i) \sim \mathcal{N}(\mu, \Sigma)$$
 (2a)

$$\mu = K(x_i, x_*)K(x_i, x_i)f_i$$
 (2b)

$$\Sigma = K(x_*, x_*) - K(x, x_*)K(x_i, x_i)^{-1}K(x_*, x_i)$$
 (2c)

The preference function f_* is then obtained by taking the posterior mean of this conditional distribution, resulting in:

$$f_* = \sum_{i=1}^{N} \alpha_i K(x_i, x_*)$$
 (3a)

$$\alpha_i = \sum_{j=1}^{N} (K(x_i, x_j) + \sigma^2 \delta_{ij})^{-1}$$
 (3b)

Lastly, since σ is the noise in our GP, it is obtained by maximizing the log likelihood of obtaining the set of seed songs:

$$\log p(f|\sigma) = -\frac{1}{2}f^{T}K^{-1}f - \frac{1}{2}\log|K| - \frac{N}{2}\log 2\pi$$
 (4)

Thus, in order to learn the preference function we must obtain a kernel K(x, y) that will serve as our similarity metric between two songs x and y. Once we have

the preference function, selecting a playlist becomes as easy as computing the preference of each song under consideration and ranking the top M.

For our project we tried two kernels, the first of which must be learned (a method called "Kernel Meta Training") and the second is a simple hamming kernel to compare our results with.

Kernel Meta Training (KMT): The idea behind KMT is to use pre-defined playlists to learn a similarity metric *K* between any two songs. In our case, we separated the 10,000 songs from the Million Song Subset into a kernel-meta-training set, a seed set, and a test set. The test set consists of the set of "new" songs that will be considered for the playlist generation. The KMT set constitutes the largest fraction of our dataset and is divided further into subsets based on some feature in order to train K. In our case, we chose to paritition the KMT set by **genre**. In contrast, the seed set is usually very small, containing as little as one song. Thus, the advantage to kernel-meta-training is that we can effectively meta-train on a much larger sample, allowing us to obtain a more accurate similarity metric.

The kernel we chose to learn was inspired by the one used in Zheng et. al. [1], and defined as follows:

$$K(x,y) = \sum_{n=1}^{N_{\psi}} \beta_n \psi_n(x,y)$$
 (5)

where ψ_n is a family of Mercer kernels indexed by n, and N_{ψ} is defined later. In order to describe ψ_n in the least confusing manner, we first make an observation about the timbre feature. For the first 4 features, a direct comparison can be made between any of these features (i.e. two songs have similar tempos if they fall within the same increment of 20 BPM). However, the timbre feature was processed into a still fairly large matrix of dimensions 12×200 . To compare two timbre features x_5 and y_5 , we take the frobeniums norm of the difference between the two timbre matrices:

$$Norm(x_5, y_5) = \text{Frobenius Norm}(x_5 - y_5)$$
 (6)

Two timbre matrices are then declared similar if their norm is below a threshold value, which we computed by taking the average norm of all tuples in the premade playlists. Now we are ready to define ψ_n :

$$\psi_n(x,y) = \begin{cases} 1 & \text{if } a_{nl} = 0 \text{ or } x_l = y_l \ \forall l < 5 \text{ and} \\ & Norm(x_5, y_5) < \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

where the vector a is the binary representation of the integer n. To understand this more intuitively, a can be

thought of as a mask that allows us to compare a subset of features at a time. In other words, ψ_n evaluates to 1 only when the components of x and y are exactly equal (or less than a threshold) whenever the component of a is 1. Since we have 5 features, we therefore have a total of 2^5 subsets of features to consider, and so n ranges from 0 to $N_{\psi}=2^5$.

Finally, to train K we want to solve for the coefficients β_n . We do so by minimizing the cost function:

$$\beta_n = \arg\min \frac{1}{2} \sum_{i,j=1}^{N} (\hat{K}_{ij} - \sum_{n=1}^{N_{\psi}} \beta_n \psi_n(x_i, x_j))^2$$
 (8)

where \hat{K}_{ij} is the empirical covariance given by

$$\hat{K}_{ij} = \begin{cases} \frac{1}{\text{# genres}} = \frac{1}{9} & \text{if songs } x_i, x_j \text{ are in same genre} \\ 0 & \text{otherwise} \end{cases}$$
(9)

Generating the seed, test, and KMT sets:

Our KMT set consists of 7,000 randomly picked songs from the Million Song Subset. To generate a seed set of size N from the remaining 3,000 songs, we first randomly selected a single track to serve as the initial seed. We then compared its features to other songs in the remaining set. If the initial seed song s_0 either shares an artist with a second song s, has 3 identical, non-timbre features, or has 2 identical non-timbre features and the timbre features are "similar" (i.e. Norm < threshold), then s_0 and s are declared "similar". However, in order to accurately mimic user behavior, we also introduce randomness into our seed generation. For every song s we compare to s_0 , we also generate a random real number $random \in [0,1]$. If s and s_0 are similar and random ≥ 0.3 , s is added to the seed set. If s and s_0 are dissimilar and random < 0.3, s is also added to the seed set. Finally, the remaining 3,000 - N songs comprise the test set.

Hamming Kernel: To compare the results we obtain with the KMT method, we also applied a simple Hamming kernel (no training required) to GPR:

$$K_{ham}(x,y) = \sum_{i=1}^{4} 1\{x_i = y_i\} + 1\{Norm(x_5, y_5) < \text{threshold}\}$$
(10)

This Kernel simply computes the number of matches between the feature vectors (and for timbre, it compares whether the two matrices are similar). **SVM + HMM:** Our desire to give a more reasonable treatment to the timbre feature led us to explore a combined SVM and HMM model. In the case of having a very small seed set, training an HMM on a set that can contain as little as a single song does not make much sense. However, it is reasonable to suppose that over time a user may have accumulated a large playlist of preferential songs and also rejects other songs in the process (i.e. from a recommended playlist on Spotify, the user only chooses a fraction of the songs; or the user either likes or skips a song on Pandora). In this case we have two training sets: a set of "likes" and a set of "dislikes". We can then train an HMM on each of these sets and the log likelihood of a training song under each HMM model become the timbre features for the SVM.

Generating the Training Sets: In order to apply SVM we must first separate our training data into two sets: one with the label "like" and the other with "dislike". We denote these sets as S_L and S_D , respectively.

To obtain these playlists, we selected an initial seed song s_0 and compared its features to other songs in the dataset. If a song s has 3 non-timbre features in common with s_0 , then s is similar to s_0 . We also added the same random component as with generating the seed set for GPR. Thus the same rules apply for adding s to S_L or S_D as for GPR.

Pre-processing Timbre: To make the size of the timbre matrices uniform across all training songs, we averaged consecutive columns to obtain a resulting "compressed" timbre matrix of dimension 12×200 . I.e. If the original track had N=612 segments, then the first 199 columns of the new matrix will consist of averaging $3=(\text{int})\frac{612}{200}$ consecutive columns. The last column will contain the average of the last 15 columns.

Gaussian HMM: Since MFCCs do not take on discrete values, we model the emission probability at each state by a multivariate Gaussian, where μ has dimensions 12×1 and the covariance matrix has dimension 12×12 .

We used a HMM toolkit for Matlab to train the HMM [6]. We experimented with different numbers of hidden states, and picked the state numbers $L_{hidden} = 6$ and $D_{hidden} = 7$ that maximized the log likelihoods of the training sets S_L and S_D . We then obtained the timbre features for a song x by computing the log likelihoods $\log p(x|HMM_L)$ and $\log p(x|HMM_D)$.

V. RESULTS

GPR: We tested both the KMT and hamming kernel on two seed sets, one containing a single seed and the other containing 3 seeds. The results are tabulated

below:

Table 2: *Top 5 Songs for GPR + KMT using 1 Seed*

	Title	Artist	Genre	Decade	Tempo	Loudness
Seed	House of Pain	Van Halen	Rock	1980	200	-10
	Tell Your Momma Come	Black Eyed Peas	Pop	2000	200	-5
	October	U2	Rock	1980	80	-5
Playlist	Goin' To The River	Alice Cooper	Rock	1980	140	-10
	Don't Waste My Time	John Mayall	Pop	1970	200	-10
	On The Road Again	The Sonics	Pop	1960	140	-10

Table 3: *Top 5 Songs for GPR + KMT using 3 Seeds*

	Title	Artist	Genre	Decade	Tempo	Loudness
	House of Pain	Van Halen	Rock	1980	200	-10
Seeds	Harajuku Girls	Gwen Stefani	Pop	2000	100	-5
	Ease	Public Image Ltd	Rock	1980	180	-10
Playlist	Performance	Happy Mondays	Rock	1980	100	-5
	Laser Show	Fountains of Wayne	Rock	1990	100	-5
	Looking for A Kiss	Sex Pistols	Rock	1970	120	-5
	The End	My Chemical Romance	Pop	2000	80	-5
	Till We Ain't Strangers Anymore	Bon Jovi	Metal	2000	140	-5

Table 4: Top 5 Songs for GPR + Hamming using 1 Seed

	Title	Artist	Genre	Decade	Tempo	Loudness
Seed	House of Pain	Van Halen	Rock	1980	200	-10
	Out Of This World	Black Flag	Rock	1980	200	-10
	Don't Waste My Time	John Mayall	Pop	1970	200	-10
Playlist	Stumble	R.E.M.	Rock	1980	140	-10
	Still a Fool	Groundhogs	Pop	1960	120	-10
	Goin' to the River	Alice Cooper	Rock	1980	140	-10

Table 5: Top 5 Songs for GPR + Hamming using 3 Seeds

	Title	Artist	Genre	Decade	Tempo	Loudness
	House of Pain	Van Halen	Rock	1980	200	-10
Seeds	Harajuku Girls	Gwen Stefani	Pop	2000	100	-5
	Ease	Public Image Ltd	Rock	1980	180	-10
Playlist	You	Radiohead	Rock	2000	100	-10
	Start Me Up	The Rolling Stones	Pop	1980	120	0
	True Nature	Jane's Addiction	Rock	2000	80	0
	Armageddon's Raid	Belphegor	Metal	2000	100	0
	Final Straw	R.E.M.	Rock	2000	100	-5

SVM + **HMM** In order to generates results with SVM+HMM that can be comparable to those using GPR, we generated the sets S_L and S_D based on the same initial seed s_0 used to generate the seed sets for GPR. Thus, although S_L constitutes a much larger set, the idea is that most of the songs in S_L are "similar" to s_0 , and therefore to all the songs in the seed sets.

We trained our SVM and HMM on a set S_L of 187 songs and a set S_D of 416 songs. We then applied our trained SVM + HMM model to classify the top 20 ranking songs in each of the four GPR runs (GPR+KMT and GPR+Hamming with 1 seed, GPR+KMT and GPR+Hamming with 3 seeds). Omitting overlaps, this meant testing on a set of 71 songs. Of these, only 3 were labelled as "like", and these songs are listed in the table below:

Table 6: Classification Using SVM+HMM

Title	Artist	Genre	Decade	Tempo	Loudness
Performance	Happy Mondays	Rock	1980	100	-5
Darling, I want to Destroy You	AFI	Rock	2000	140	0
Stumble	R.E.M.	Rock	1980	140	-10

VI. DISCUSSION

GPR: To compare the performance of different learning algorithms, we used standard collaborative filtering metric to score the generated playlist in each trial. The score of the playlist from trial j is defined as:

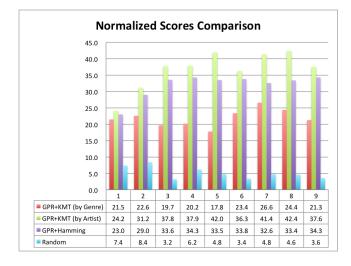
$$R_j = \sum_{i=1}^{N_j} \frac{t_{ij}}{2^{(i-1)/(\beta-1)}} \tag{11}$$

Where $t_{ij} = 1$ if ith song in the generated playlist is in pre-made playlist in trial j; 0 otherwise. Beta determines how fast user interest decays, and is set to 10. N_j is the number of songs in the generated playlist. The score is then summed and normalized as

$$R = 100 \sum_{j=1}^{1000} R_j / \sum_{j=1}^{1000} R_j^{max}$$
 (12)

Where R_j^{max} is the perfect score in trial j. R = 100 corresponds to perfect prediction and larger R values indicate better performance.

Figure 1: Histogram of Scores of the Various Methods



All variations of GPR significantly outperform the randomly generated playlist. However, GPR+KMT does not consistently win GPR+Hamming. How we make the pre-defined playlists is important. Treating all songs in the same genre as similar tends to over-generalize and results in worse predictions than

Hamming kernel. On the other hand, treating songs from the same artist as pre-defined playlists results in much better predictions of the user prefenrece. Based on the observation above, KMT works well only with well designed pre-defined playlists, and can constantly outperform GPR+Hamming.

SVM + HMM: Although we did not have enough time to write a scoring algorithm for SVM, we note that 2 of the 3 songs in Table 6 were ranked at top 10 by GPR+ KMT and GPR+ Hamming. Thus the two methods do appear to overlap in terms of which songs they predict the user might like. In general, however, SVM and GPR are very different approaches to playlist generation and therefore difficult to compare. GPR is good for very few seeds, but has an extremely simplistic way of dealing with timbre. SVM depends on their being two large training sets S_L and S_D , allowing us to train an HMM for timbre for each of these sets. Thus each method has their own unique advantages and determining which method is best depends on the situation.

VII. Conclusion/Future Work

There are a lot of interesting avenues to explore from here. We could, for example, train our SVM + HMM model using the more intricate kernel defined in equation 5 rather than the linear kernel which we used for this project. We could also expand upon the feature set by including lyrical features—i.e. judging the meaning/content of the music rather than only its acoustic features.

REFERENCES

- [1] Zheng et. al. (1996). Learning a Gaussian Process Prior for Automatically Generating Music Playlists
- [2] Sandler et. al. (2005). Timbre Models for Analysis and Retrieval of Music Signals IEEE Transactions on Multimedia, Vol 7, No. 6
- [3] Zhang, X., Ras, Z., Dardzinska, A. (2005). Discriminant Feature Analysis for Music Timbre Recognition and Automatic Indexing
- [4] Platt, J. C. (2006). Auto playlist generation with multiple seed songs
- 5] Ebden, M. (2008). Gaussian Processes for Regression: A Quick Introduction
- [6] Murphy, K. (1998). Hidden Markov Model (HMM) Toolbox for Matlab