Analyzing Grammy Data using Topological Data Analysis

Michael Mendelsohn
Matthew Elgart
Matthew Faw

Math 412 - Topology With Applications
John Harer
Duke University

May 4, 2017

Contents

1 Introduction **1**

2 Related Work **1**

3 Data Collection **1**
3.1 Collecting Grammy Data . . . 1
3.2 Interfacing with Spotify Web API . . . 1
3.3 Constructing Artist Catalogs . . . 2
3.4 Serialization . . . 2
3.4.1 CSV . . . 2
3.4.2 Pickle . . . 2

4 Methods **2**
4.1 Artist Risk Analysis . . . 2
4.2 Delay Reconstruction . . . 3

5 Analysis **3**
5.1 Record of the Year Time Series . . . 3
5.2 Artist analysis . . . 3
5.2.1 Album granularity analysis . . . 3
5.2.2 Comparing artists analysis . . . 4

6 Extensions **6**

A Data, Plots, and Code **6**

1 Introduction

Winning a Grammy Award is one of the most prestigious honors a musician can achieve. We wanted to find out whether Grammy-winning songs share common or periodic attributes, and whether artists who win Grammies are more likely to attempt to make multiple styles (measured by topologically significant features) of music than their non-Grammy winning counterparts. In an attempt to answer these questions, we used Spotify's Spotipy API to acquire our dataset, and topological data analysis techniques in order to answer our questions.

2 Related Work

There have been attempts to use regression to predict the winners of entertainment awards including Grammys. Spotify has been successful at predicting Grammy winners by using big data techniques[1]. Others have investigated using regression techniques to predict Grammy winners and losers[1]. This attempt was not successful, but the features considered in this regression analysis are primarily related to popularity of the songs on Billboard charts. To our knowledge, other attempts to analyze Grammy data using Spotify feature vectors have not been made. While a regression analysis that includes these Spotify features could be of interest, we will only explore topological data analysis techniques in this paper.

As discussed in class, Chris Tralie has used TDA to perform excellent analysis into the periodicity of time-series data[2]. His techniques involved using delay reconstruction to create a high-dimensional sliding window embedding of the time series. He then performed PCA to project this down into lower dimensions, and conducted 1-D persistence on it. A significant 1-cycle would imply periodicity in the data.

3 Data Collection

3.1 Collecting Grammy Data

The Grammy award data used in this project is listed at `https://www.grammy.com/awards`. To collect this data, we used the Selenium[2] python library to crawl the Grammy awards website and thus obtain a dataset with which we could perform our analysis.

For each Grammy award, we collected four fields: year the award was given, name of the award, the work that won the Grammy, the award recipients.

3.2 Interfacing with Spotify Web API

In order to perform data analysis on Grammy-winning songs, we needed to construct features that represent the songs. Spotify presents an API to obtain features for songs, but to obtain these features, we needed to extract song names and artist names from the scraped Grammy entries. Because song names on the Grammy website often did not *exactly* match song names on Spotify, and because artist names were often not easily extractable from the award recipients list, programmatic parsing of the scraped data was often insufficient to obtain feature vectors for all Grammy winning songs. Thus, some manual data manipulation was necessary. In order to minimize the amount of manual data manipulation, we calculated the number of query misses for each Grammy category. Because the `Record of the Year` category had a low number of query misses, and because the category has been a category from the beginning of the Grammies, we decided to restrict our analysis to this category.

[1] http://www.funktasy.com/big- data-predicts-grammy-award-winners/
[2] `http://selenium-python.readthedocs.io`

Once we parsed the scraped Grammy data into valid strings of artist/song pairs, we needed to make the call to Spotify's web API in order to retrieve the corresponding feature vectors[3] (of the following categories, rated from 0 to 1: danceability, energy, speechiness, acousticness, instrumentalness, liveness, and valence). To help with this, we used Spotify's recommended Python module, spotipy, to facilitate the API call. This made it easy to obtain a feature vector given a valid artist/song pair, and so all that was required was to iterate through all pairs in a relevant subset of songs (Grammy category, artist catalogue), and make the API call to construct feature vectors for each one.

3.3 Constructing Artist Catalogs

In order to perform more comprehensive analysis on the Grammy-winning artists, we collected data on every album the artists released to Spotify. For each Grammy-winning artist, we collected feature vectors for every song on every album the artist released on Spotify. Album data was again obtained using Spotify's spotipy API[4].

3.4 Serialization

Because obtaining the Grammy and Spotify data required making a large number of HTTP requests, running our data collection scripts often took several minutes to run. So we only had to run these scripts once, we serialized the data obtained from these queries into local files that we could manipulate and later load to perform analysis. We used two different serialization strategies – outputting to CSV and JSON file formats.

3.4.1 CSV

Comma separated value (CSV) files have the advantage of being easy to read and manipulate. Since our scraped Grammy dataset needed to be easily modifiable, we chose to serialize the data into CSV files.

3.4.2 Pickle

While CSV files are easy to manipulate and read, serialization of generalized objects is not always simple. Additionally, not all saved objects need to be easily read or manipulated manually.

Luckily, the `pickle`[5] python library provides simple object serialization to JSON files. Though these files are not always human-readible, they allow simple save/load functionality so that we were able to save and load arbitrary objects while performing analysis.

4 Methods

4.1 Artist Risk Analysis

In order to determine an artist's willingness to take risks, we first filtered our dataset for all of an artist's albums. Then, for each album, we filtered out invalid values, where information on a songs was missing. We then conducted 3-component PCA on the feature vectors of the songs in the album. We chose 3 components so that we could visualize the data. For a more complete analysis, we could have used the Kaiser criterion[6] to determine the number of components to keep. Once PCA was computed, we took both the PCA dataset and the dataset before PCA and computed 0, 1, and 2-D persistence diagrams.

[3] `https://developer.spotify.com/web-api/get-audio-features/`
[4] `https://spotipy.readthedocs.io/en/latest/`
[5] `https://docs.python.org/2/library/pickle.html`
[6] `http://www.uta.edu/faculty/sawasthi/Statistics/stfacan.html`

In addition to computing the persistence diagrams on each album, we computed them on the collection of all of an artist's songs as well.

Due to the large number of artists that we had collected data on, we decided to select three artists who one record of the year to conduct the above analysis on (Coldplay, Eric Clapton, and U2). For a baseline, we selected artists that Spotify recommended as related artists to the ones we had selected, and who had not won Record of the Year (Buddy Guy, Derek & The Dominos, Keane, R.E.M., Snow Patrol, and The Police).

4.2 Delay Reconstruction

One of the ways that we wanted to explore the data using TDA was to perform similar analysis to Chris' project: using delay reconstruction, create a sliding window embedding of the data, and then perform 1-D persistence to test for periodicity. The Grammy category of Record of the Year lent itself very well to this form of analysis; as mentioned previously, it has a comparatively long history, and high coverage from Spotify's database. Of course, the yearly nature of the award naturally lends it to time series data as well, which made the delay reconstruction periodicity testing even more appealing.

When performing delay reconstruction, we created the window by concatenating a sequence of vectors of some variable length to create a high dimensional window vector. We then used PCA to project this data down into lower dimensional data (we looked at both 2 and 3 dimensional data, for vizualization purposes). We therefore sweeped through parameters for window size as well as number of components for PCA, using Ripser code to see which parameter values gave the most significant 1-D persistence (if any).

5 Analysis

5.1 Record of the Year Time Series

We found the most significant 1-D persistence when using a sliding window of size 32 (i.e., concatenating a vector in the time series data with the following 31 vectors to construct the window). Shown below are 1-D persistence plots for both PCA projection into 2 and 3 dimensions, as well as scatter plots of the projected data. We can see the 1-D persistence in these scatter plots, which shows that the characteristics of songs winning Record of the Year (as measured by Spotify's feature assignment) do indeed show some kind of periodicity!

Unfortunately the raising and lowering of dimensions makes it difficult to see along which specific dimensions the data has the most significant periodicity. However, it is certainly interesting that we were able to find 1-D persistence in the time series of record of the year- this implies that the records recognized as best have not only changed, but have in fact shown periodic trends, over time.

5.2 Artist analysis

5.2.1 Album granularity analysis

Since winning a Grammy can be such a special moment in an artist's career, one might wonder how an artist's albums change over time, and if winning a Grammy causes an artist to be more or less risky in the work they produce after winning.

We investigate the feasibility of using persistence to study how an artist's albums change over time. Figure 2c displays persistence diagrams obtained for three of U2's albums. The leftmost figures display 0D persistence for an album U2 released prior to winning a Grammy for Record of the Year, the middle figures displays 0D persistence of the album containing a song which won Record of the Year, and the rightmost figures displays an album U2 released after winning Record of the Year. The persistence

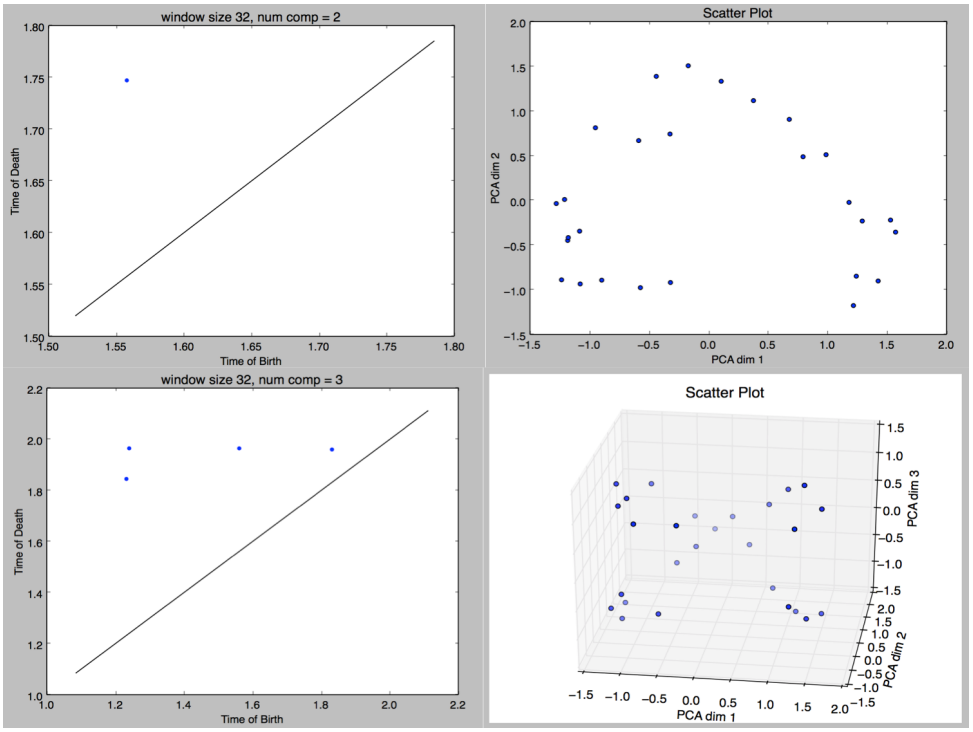


Figure 1: The leftmost figures are the Sliding Window Persistence diagrams produced from the data plotted on the rightmost figures.

diagrams on the top row are produced by running PCA with 3 components. Persistence diagrams on the bottom row are produced without PCA.

The main complication with extracting meaningful analysis from these plots is the noise in the persistence diagrams. It is not obvious on what scale the persistence diagrams should be considered. Thus, there is no clear way to objectively count the number of 0D components.

If we take the closely-clustered features in the diagrams as a single feature, it seems that the persistence diagrams with and without PCA appear to have the same number of features. The first album (leftmost) seems to have only a single component. The album with the Grammy-winning song (middle) seems to have 3 components. The last album (rightmost) seems to have two components.

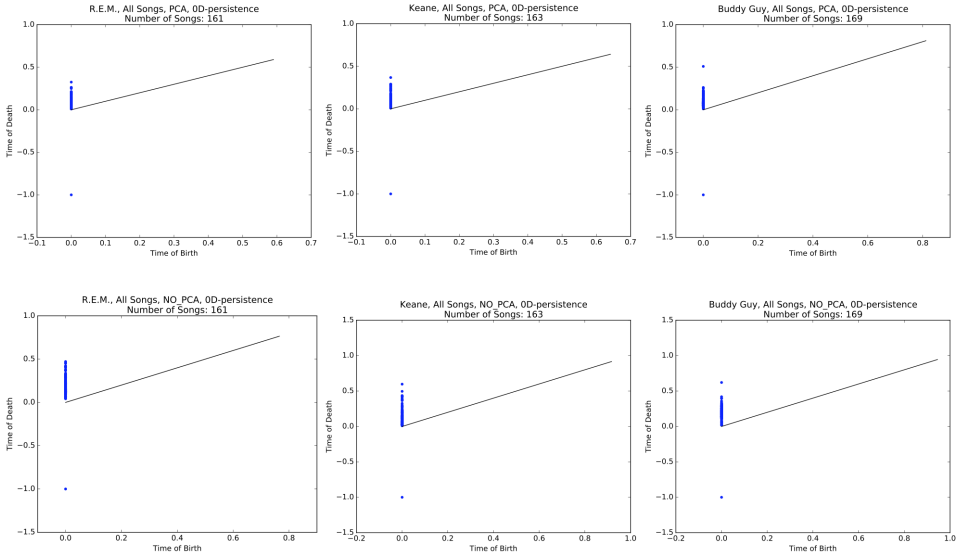
While no conclusions can be definitively drawn from this data, it appears that, for this artist, the greatest number of 0D components occurs for the album containing the Grammy-winning song. Perhaps the number of 0D features could be used to predict whether a particular song might win a Grammy. Or perhaps, more simply, given an artist's entire catalog, the number of 0D features could be used to predict which album is most likely to be critically-acclaimed.

5.2.2 Comparing artists analysis

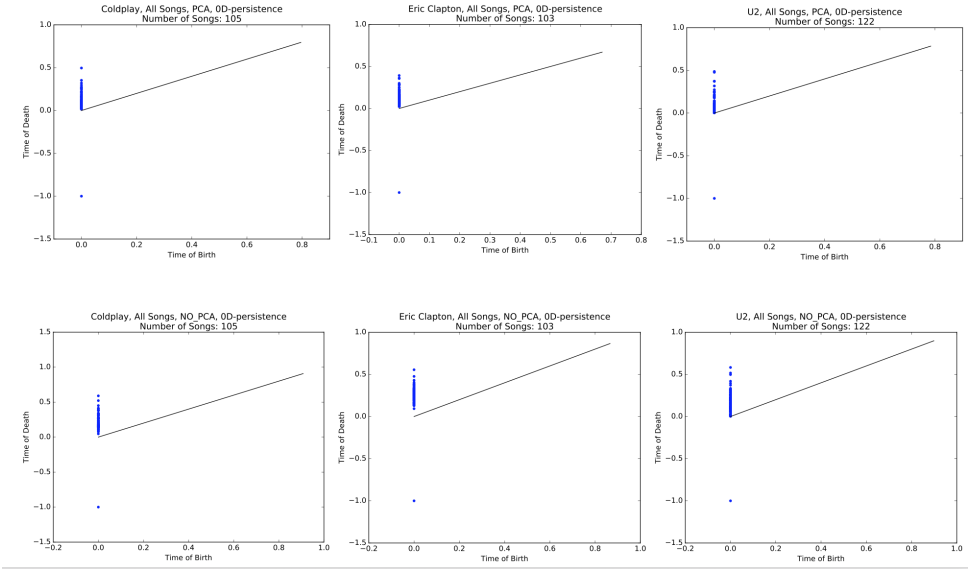
We are comparing Grammy-winning artists against Spotify's related artists. The question we are planning on asking here is – are Grammy-winning artists more *risky* than Spotify's related artists?

Figure 2a show the 0-D persistence from the Non-Grammy winning artists on all of their songs. Higher points on the persistence diagram indicate where the artist "took a risk" and was willing to produce different types of songs. Among the Non-Grammy winning artists, R.E.M. had one cluster of songs, while Keane had two fairly closely related groups, and Buddy Guy had two distinct types of songs. However, it is difficult to conclusively determine whether the plots signify significant differences.

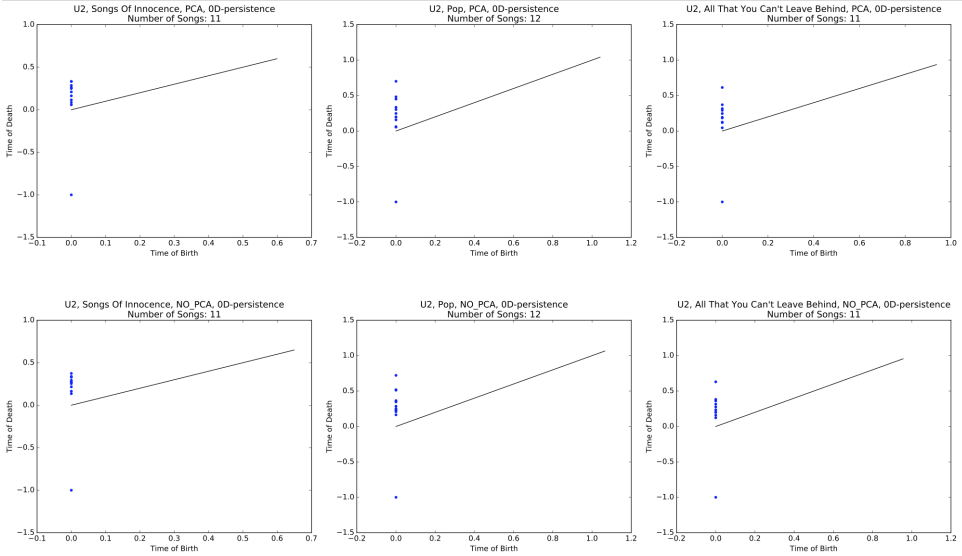
Figure 2b has 0-D persistence for Grammy winning artists. Coldplay, Eric Clapton, and U2 all have multiple groups, however, the number of groups and difference between the groups varies.



(a) 0D Persistence diagrams for some of the baseline artists' entire music catalog



(b) 0D Persistence diagrams for all of Coldplay, Eric Clapton, and U2's entire music catalog



(c) 0D Persistence diagrams for U2's albums released before, at, and after their Grammy-winning song

Figure 2: Persistence diagrams for the artist analysis

While the Grammy winning artists that we analyzed seem to have more separated groups of songs than their Non-Grammy winning counterparts, the difference between the variance in the two groups of artists is rather small. It is difficult to conclude whether or not the Grammy winning artists are more or less likely to take risks by making different types of songs.

6 Extensions

One of the nice aspects of this project is how applicable the TDA methods we used are to other datasets, and how many options that we had when narrowing down our final methods. The result of these options is that there are several ways in which we could extend our project. For example, we could easily apply our methods to more of the many Grammy categories, and see if our methods could give us any insights for those. Similarly, we could perform our analysis on other existing datasets, such as Emmies, Tonys, or other entertainment awards.

There are options for extension in our current project workflow as well. The simplest would be to explore different PCA dimensions: 2D and 3D data is handy for visualization, but otherwise arbitrary. It would be interesting to see if there are any other insights to be gained if we project the data less. As previously explained, the Kaiser criterion is one method that could have been used to determine how many principle components should be used. This would make it easier to expand our analysis and see if it is possible to predict the characteristics of *future* Grammy winners as well.

Another extension we could add is a computational analysis of the persistence diagrams of an artist. For example, we could use machine learning to predict whether or not a persistence diagram came from a Grammy winning or Non-Grammy winning artist. This might also be useful for predicting future Grammy winning songs and albums.

Finally, it would be interesting to make use of statistical methods as well. Many of the questions that we use TDA to answer share an overlap with statistics. As a result, we could perform both, and compare how effectively the two methods compare.

References

[1] D. E. Peacock and G. Hu. "Analyzing Grammy, Emmy, and Academy Awards Data Using Regression and Maximum Information Coefficient". In: *2013 Second IIAI International Conference on Advanced Applied Informatics*. Aug. 2013, pp. 74–79. DOI: 10.1109/IIAI-AAI.2013.14.

[2] Christopher J. Tralie. "High-Dimensional Geometry of Sliding Window Embeddings of Periodic Videos". In: *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*. 2016, 71:1–71:5. DOI: 10.4230/LIPIcs.SoCG.2016.71. URL: http://dx.doi.org/10.4230/LIPIcs.SoCG.2016.71.

A Data, Plots, and Code

All of our code is available at https://github.com/matthewfaw/math412-final-project.

The scraped Grammy data is available here: https://github.com/matthewfaw/math412-final-project/blob/master/data_collection/grammy_data/RECORD_OF_THE_YEAR.csv

We performed analysis on several different artists – some of the plots we produced are available here: https://github.com/matthewfaw/math412-final-project/tree/master/plots.

The GitHub repository contains all of the code we used to analyze our data. It also lists our main library dependencies and documents our main API.