Music with Machine Learning

COGS 118B Final Project:
GTZAN Dataset, Audio Analysis

Authors: Evan Eguchi , Shushruth Kallutla , Elliot Lee , Finn St John ,
Shantelle Serafin , Michael Lee

Background and Overview

For our final project for this class, we decided to analyze audio files. We are all very passionate about music, and for this reason, we knew we wanted to do some sort of analysis on music. After a long search on Kaggle, we stumbled upon the GTZAN Dataset, which is a dataset of 1000 thirty-second audio samples from different songs, along with their spectrograms. We decided to split into two teams: one to analyze the spectrogram data and one to analyze the audio files.

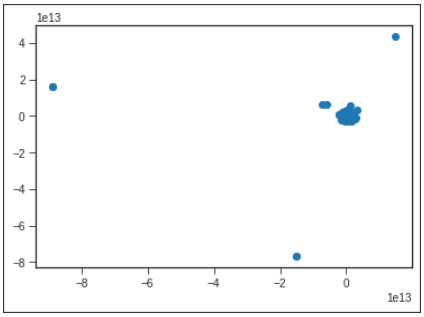
The dataset is divided into ten genre groups of 100 songs each. These genres include blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. After some preliminary exploration of the data, we found that when decoded, the .wav files were each ~660,000-length integer time series arrays. Each of these values represented 1/22,050 of a second, or 45.35 μs. The spectrogram images after processing had 73248 dimensions. Because of the high-dimensionality of this data, we thought PCA would be a great first algorithm to implement. Additionally, because K-means is a simple, but effective algorithm, we decided to implement it as well, hoping to compare our unsupervised classifications with the true genre classifications for both datasets.

Method 1: PCA on Audio Time Series

Because each audio sample was a 660,000-dimensional vector, we thought PCA's dimensionality reduction capabilities would effectively apply to our dataset of 1000 songs. We knew that the maximum dimensionality that the data could actually occupy would be at maximum the sample size. We decided to use a non-probabilistic PCA algorithm. After computing the eigenvectors and eigenvalues of the covariance matrix, and plotting the eigenvalues in sorted order, we noticed that there were five eigenvalues that accounted for the most variance in the data. For this reason, we decided to use five principal components when reconstructing the data. The PCA-space for this algorithm was a 1000-dimensional vector space of audio files.

One large issue we ran into was the limit on execution memory. Google Colaboratory only provides a certain limited amount of RAM for us to use during execution of each Jupyter Notebook and even calculating the eigenvalues and eigenvectors of the covariance matrix takes a large amount of memory. We were able to overcome this issue with some trial and error. Our final solution to this problem being to not store the covariance matrix in a variable.

Implementation of the PCA was a very similar process to what we had done previously, so it wasn't as difficult or time consuming this time. Unfortunately, after this work and after looking at the results, the PCA algorithm does not seem to be effective for this type of data. When we reconstructed the audio using the complete set of principal components, the resulting audio was not recognizable. We believe this to be due to the fact that the principal components that we found to reduce the data were still not representative of the entire dataset. Additionally, when we reconstructed our data using a reduced principal component matrix and mean songs, it would produce an overlaid sample of a few songs along with some noise.



This figure shows the data song data projected onto the vector space of the first two principal components. Clearly, there are some very strong outliers in the dataset.

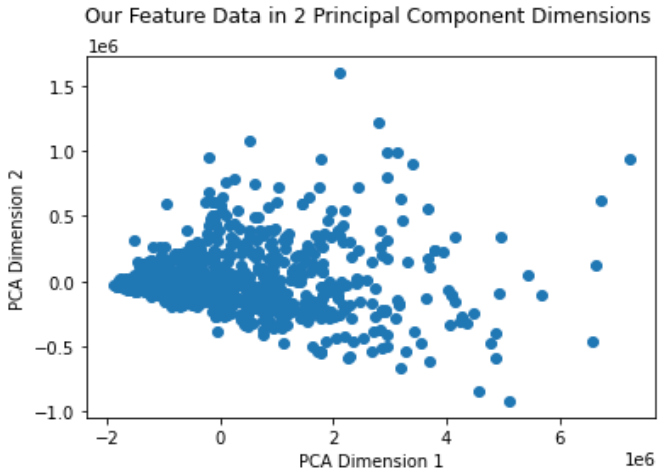
Method 2: PCA on Audio Features

Due to a lack of desired results on the raw audio data, we decided to perform PCA using features of the dataset provided to us by the dataset authors. There was an additional file included in the GTZAN Kaggle Dataset that contained audio features for every song in the dataset. These consisted of tempo, length, roloff, harmony and some distributions of MFCC variables.

After some inspection, we found that it was possible to reconstruct a playable audio file from an array of MFCCs. MFCCs are the coefficients to a short term power spectrum of sound.They are also a way to encode sound data, but conversion is lossy. We found that it is possible to take an audio file and convert it to a large amount of MFCC values. We could then take these values and partially reconstruct a song. However, the feature dataset we used gave us a mean and variance of these MFCC values. We found that it is not possible to sample from these given distributions and

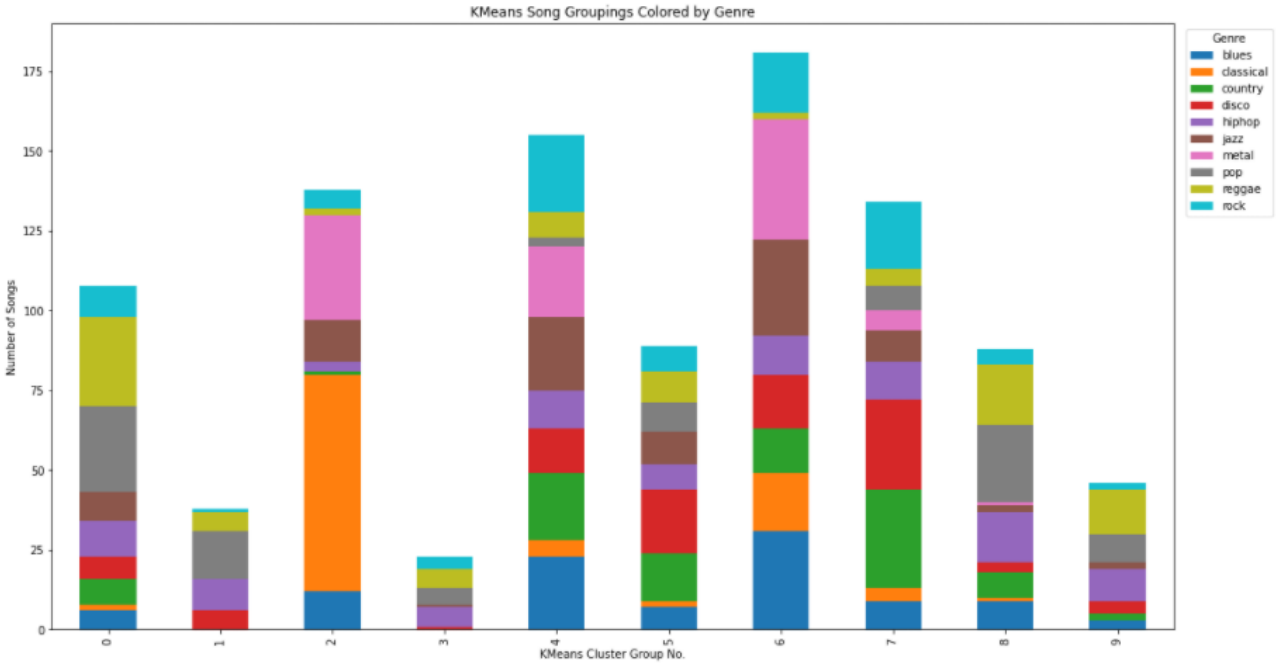
reconstruct a piece of audio. Thus, without the audio from the other dataset, it was impossible to play any of these songs.

Although we couldn't reconstruct audio from our data, we were still able to perform PCA on it. Using SKLearn's default PCA library, we found that most of the variance can be explained by the first component. Additionally, plotting the feature data in the first two principal component dimensions produces an almost heteroscedastic plot. The conclusion that we draw from this data is that features may be correlated for some songs, and much less for others. This song space that we are working in should be further reduced in order to more accurately capture certain features of a more specific data set, or more features should be included to more generally describe the components of the data.



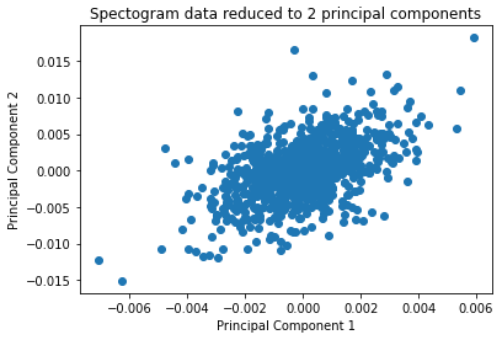
Method 3: K-Means on Audio Features

Initially we wanted to perform K-means on the dimensionality-reduced PCA data. However, we found that this data, although smaller, was not a good representation of our data. Instead, we performed K-means clustering using the provided audio features. Our K-means algorithm was implemented similarly to project 1, using Expectation Maximization and. We first calculated cluster assignment and then estimated the means of our clusters over multiple iterations. Comparing our algorithm to the SKLearn K-means algorithm, we found that the groupings were similar. The resulting groupings were not even, and they contained a large variety of genres, indicating that the algorithm did not cluster based on genre. The conclusion that we can draw from this is that songs across genres are probably more similar than we expected them to be, and additionally probably have high variability within their genre. The only genre that stood out against this conclusion was classical. Classical was consistently grouped with a large majority of its own genre.



Method 4: PCA on Spectrograms

In order to ease processing for our genre clustering on the spectrogram images, we performed dimensionality reduction using PCA. Each image of the spectrogram had dimensions (288, 432, 3); It's a 288 by 432 matrix with 3 RGB values for each pixel. Since the spectrogram had some white borders around it when imported from the Keggle, we had to process it into a matrix of (218, 336) per spectrogram. We then flatten the matrix into a singular array to be able to take the mean and find the eigenvalue/vector of the mean spectrogram. We reused some of the helper functions used in previous homework problems to help sort the eigenvalues/vectors, which was used to find the vectorization of the images minus the mean spectrogram named U. We then normalized U and found the principal components by multiplying the transpose of U and transpose of image_minus_mean. We can then recreate the original images by multiplying U and components to get Z, and adding image mean to the transpose of Z.

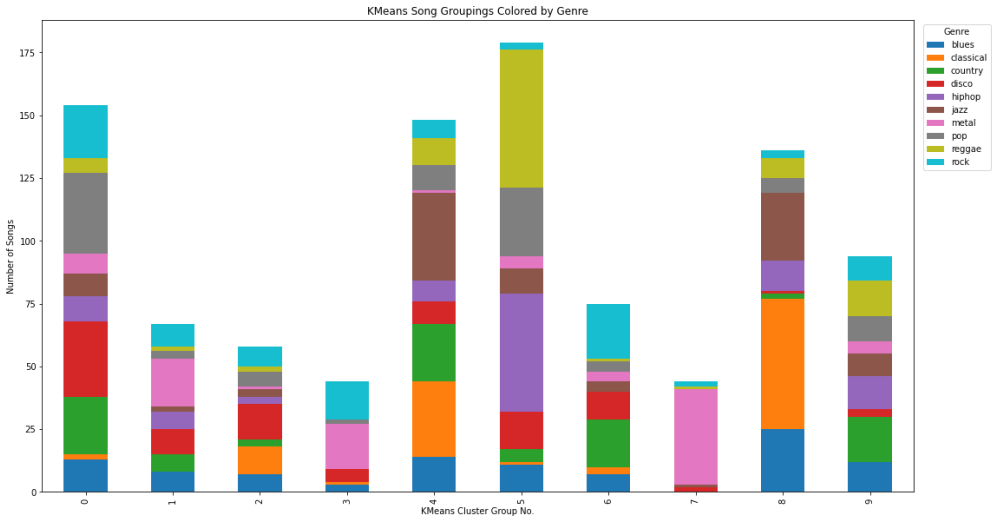


This figure shows the spectrogram data reduced to two principal components

Method 5: K-Means on Spectrograms

We decided to perform a K-Means clustering model on the reduced (100 components) spectogram dataset to separate the spectrograms into their respective genres. Utilizing the vectorized functions previously defined in class through homework assignments, our K-means function iterated many times to fine tune randomly initialized cluster centers by calculating squared distances between data points and temporarily defined cluster centers, determining rankings of each data point (which genre they belonged to), and recalculating the cluster centers through transposition and dot products of the rankings, data points, and ranking sums. Once the difference between a newly defined group of cluster centers and a previous group of cluster centers was small enough, the final cluster centers and rankings could be defined.

Looking at the results below, we can clearly see that our k-mean implementation has a lot of variability present in its classification. Although some genres show promising cluster grouping, it's not nearly enough to be called accurate. As well as the different genre sizes within the clusters, the cluster sizes themselves also fluctuate more than normal. In a perfect world we would expect a 100 song split across each cluster group (besides the jazz cluster, which has one deprecated file), but in our distribution the amount of songs is clearly different within each cluster. We predict that this distribution could have been affected by treating sound data spectrograms as normal colored images and the overall complexity of sound data in general.



Discussion

In this project, we solidified our understanding of principal component analysis and k-means clustering by applying them to audio data. Through the use of both self created and predefined functions and packages, we came to see the various steps required to carry out these machine learning models and the accuracy levels they can potentially reach with proper refinement. Looking at all of our dimensionality reduction and clustering results, clear improvements can be made in order to increase the accuracy of our intended genre grouping. In the future we would look into refining our PCA methods to raise the reduced data's overall representation of the original data. We would also expand our unsupervised machine learning model arsenal to potentially utilize more accurate and better suited models on this audio data, straying away from k-means clustering. Although k-means clustering has the capability to work, there are no doubt more optimized methods of approach for this type of clustering with audio, which we would utilize to improve accuracy. In terms of expanding our project, we would look into creating a model that could generate certain genres of music based on prior knowledge and desired parameters for the generated song. Once classification is optimized and genre clustering can be demonstrated and proved, the natural next step would be to transition that into a music generator which could create interesting and (hopefully) entertaining songs.

References

- https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
- https://en.wikipedia.org/wiki/Mel-frequency_cepstrum#:~:text=Mel%2Dfrequency%20cepstral%20coefficients%20(MFCCs,%2Da%2Dspectrum%22).
- https://musicinformationretrieval.com/mfcc.html

Links

- https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification
- https://github.com/stjohnfinn/COGS118B-Final-Project
- (youtube link here)

Contributions

- Elliot Lee
  - PCA of audio time series
  - MFCC research
  - PCA of feature data
  - Writeup
- Shantelle Serafin
  - MLE of audio time series
  - MFCC research
  - Group 1 Slides
- Finn St John
  - K-Means of feature data
  - PCA of audio time series
  - Writeup
- Michael Lee
  - PCA on spectrogram
  - Writeup
- Shushruth Kallutla
  - Co-wrote K-Means functions
  - PCA on spectogram
  - Group 2 Slides