Music Classification with Supervised Machine Learning

Constance Wang

Abstract

In this problem, supervised machine learning is studied using principal component analysis and linear discriminant analysis to perform classification. Music from different artists and different genres were categorized using this workflow technique. This technique also utilizes spectrograms to best characterize the music pieces. Most music was able to be characterized and differentiated, however, the model fails to recognize music from very similar artists.

Section I – Introduction and Overview

Machine learning is a prominent method of data analysis in the field of data science. In general, there are two distinct categories of machine learning: supervised and unsupervised machine learning. Supervised machine learning is used for regression and classification of labeled data. Unsupervised machine learning uses unlabeled data to draw conclusions. In this problem, supervised machine learning is used for classification, or separating data into different, labeled categories.

The labeled categories used in this problem are music genres, and the data is 5-second clips of music. These different music samples come either from artists from different genres (test 1), different artists of the same genre (test 2), or different genres (test 3). Supervised machine learning will be used on some training data to build a model that can be used to differentiate the different genres/artists. The effectiveness of this model can be tested using sets of testing data composed of different clips (but still from the same genre/artist) to make sure the model is an effective classifier.

Section II – Theoretical Background

To effectively classify the different music artists/genres, an understanding of the different steps used to build a supervised machine learning model is necessary. In general there are four steps involved:

1. Prepare the data to find the most important information
2. Principal component analysis (PCA) to determine the most important features
3. Linear discriminant analysis (LDA) to determine a threshold to divide between categories
4. Test the model using test data to verify the effectiveness of the classifier

Spectrograms for data preparation

To best represent the most important information in the data, spectrograms of the music samples were taken. Spectrograms take consecutive Gabor transforms across the length of a signal. To do this, there is a particular "window" function that is shifted by some value (often called $\Delta\tau$) across the length of the signal to collect frequencies across the length of a signal. For this problem, the "window" function used is a Gaussian, which filters out frequencies outside of the Gaussian region:

$$windowfunction = e^{-a((t-\tau)^2)} \quad \text{Gaussian Function (1)}$$

Spectrograms are particularly useful for signal analysis, because the most important frequencies are recorded with larger magnitude at each τ value. This is particularly useful for principal component analysis to work most effectively.

Principal Component Analysis to Determine Important Features

Principal component analysis (PCA) is a statistical tool that is often used to determine which components of a set of data are most important. This is important in this problem to find a way to represent the most important information in the data. PCA utilizes singular value decomposition (SVD), which is given below in Equation 2:

$$A = U\Sigma V^* \qquad \text{Equation 2}$$

The Σ term contains the singular values, which represent the magnitude or how important a particular component is. In practice, not all singular values are used, and the number of singular values uses is typically known as the number of features (how many features are used to represent the data). These features represent the number of modes (principal component modes) used. This can be an important parameter to vary to help with overfitting of the model. The V^* term is used to represent projections onto the determined number of modes, which is then used in the next step.

Linear Discriminate Analysis to Determine Categories

After projecting onto the determined number of modes, the linear discriminate analysis is used to project the high-dimensional data even further. The number of modes represents how many features are necessary to represent the data. To conduct simple classification analysis, this data needs to be instead represented with one value. This means the data must be further projected onto a line, so that values above or below some thresholds distinguishes them from different categories. Determining this what line to project on is typically done by minimizing variance within the category, and maximizing variance between categories. These variances are known as class variances and are defined below. These can be satisfied by solving the generalized eigenvalue problem:

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w} \qquad \text{Equation 3}$$

The maximum eigenvalue of that satisfies this equation will be the direction of which the best projection will occur. This means that projecting the data onto this direction will help distinguish the categories as best possible. Threshold(s) can then be determined (in this problem, a threshold that created an equal number of incorrect guesses between two categories was chosen) to split the data between different categories.

Test the Model with Test Data

One issue with constructing a model without test data is that overfitting can easily occur (you could create a situation where you perfectly predict each category the data belongs in). Test data is important to verify the model works on a wider range of applications. To test the model, test data simply has to be in the same format as the original data, and go through steps 1-3 after the model is created (i.e. the data is transformed into a spectrogram, projected on the most important features, and categorized using the threshold values determined). This will test how robust the machine learning model is.

Section III – Algorithm Implementation and Development

There are three different tests that were given in the problem description, however, the process used to obtain the model for the data remained the same. There were three different categories in each of the tests that would split the music samples. The implementation of the four steps listed in Section II are given by different functions created in the code (steps 2 and 3 were combined).

Function `spectrogram_matrix()`
This function focused on creating the spectrogram for each of the music files. Each music file contains a recording of either a single artist (such as in Test 1 and Test 2) or a single genre (Test 3). It has some number of different 5-second music samples, all one after the other. All samples were original recorded at 44100 Hz, however this sample rate can cause problems when trying to analyze and produce these spectrograms (MatLab cannot easily handle such a large amount of data).

The first step was to break down each music file into the separate 5-second clips (which at first are the different rows of the matrix). The next step, however, was to reduce the number of samples taken by only taking every 3 sample points from the original sample. This helps cut down the number of samples taken, and most frequencies are still captured. To store the spectrograms, the width and $\Delta\tau$ needed to be initialized (which were determined empirically after plotting a series of music samples from different artists) to create the Gabor transform function (`filter` in the code). This was then multiplied by the song signal and the Fourier Transform was taken to determine frequency character at the specific τ value. However, spectrograms are typically stored as matrices. In this matrix format, this means that PCA and LDA cannot be used, so each time a spectrogram was generated from a music sample, it was converted into a column vector. These column vectors were stored in a matrix to be used in the next function.

Function `trainer()`
This function completed PCA and LDA on the music files. For all the tests, there were three different categories studied. Therefore, this function takes in three spectrogram matrices (one for each of the different categories), combines them in a matrix, and computes PCA and LDA. To compute PCA, the SVD was taken using the `svd()` command in MatLab. The number of features that were selected varied between each test, and therefore was determined empirically depending on the result of the training model (test 1 used first 30, test 2 and test 3 both used 50). First, projection onto the principal components was done by $\Sigma * V^*$ as discussed in Section II. V^* is simply V^T since all the numbers are real in this case. Note also that U is stored because these are the components that will be projected on with the test data.

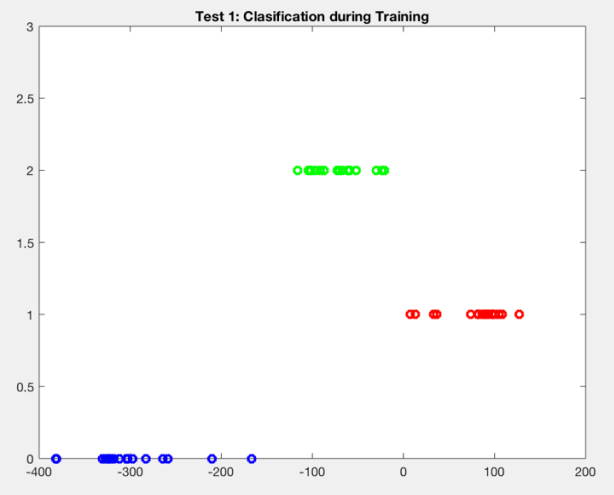
After projection onto the components, these data can be separated once more to compute the LDA. The class variances were computed using the formulas given in Section II. Solving the general eigenvalue problem was done using the `eig()` command. The location of the largest eigenvalue was determined so that the data could then be further projected onto this direction (which was normalized so that magnitude is not an issue) so each sample is represented with one number. Finally, the threshold was determined. Two threshold lines were determined to distinguish between the three categories. The data is sorted, which means that if there is overlap between classes, one way to create the threshold is to ensure that there will be a similar number of errors occurring in either class. This is done by comparing the largest value of one category with the smallest value of another category (assuming the first has a larger magnitude for the average of all samples). Thus, an average can be found between these two values so that this overlap region is properly accounted for, giving either category an equal chance of error.

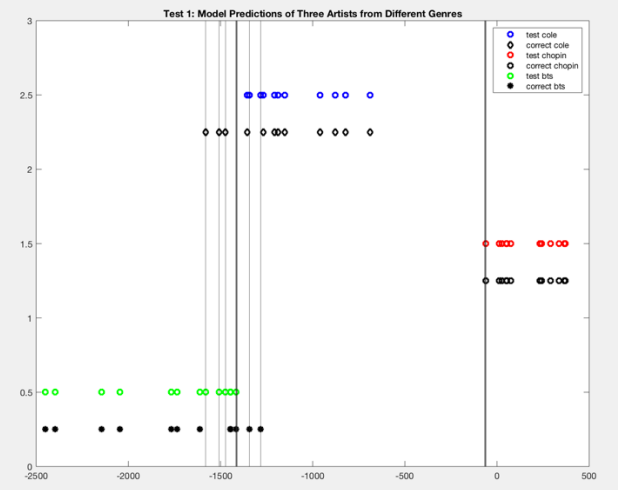
Function `tester()`
In this final function, the code is tested. The testing data is in a similar format as all the other data, and is labeled very simply (all data from the different genres are one after the other). Therefore, the testing data simply needs to be run through the `spectrogram_matrix()` command, and then

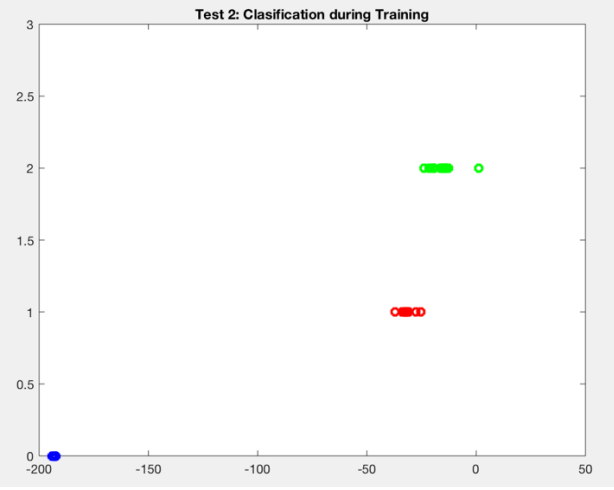
projected onto the principal components U and then onto the LDA. Conditional statements were set up so that the test labels could be compared with the actual correct labels, which is saved into a matrix for use in the graphics.

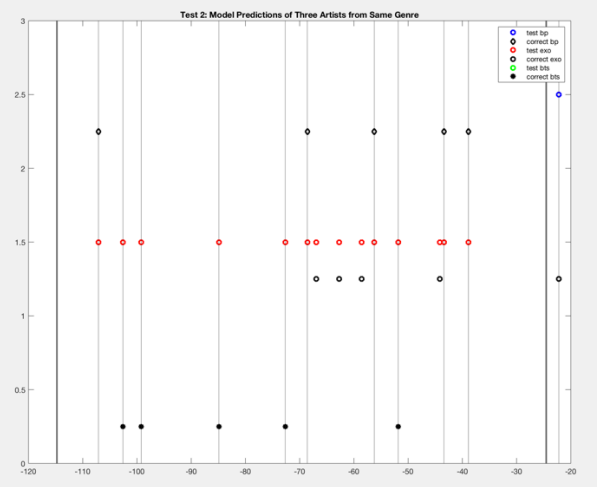
Section IV – Experimental Results

Each of the tests yielded different results depending on how well the model could not only separate the different artists/genres, but also recognize patterns of the different artists/genres (to correctly label and categorize new music samples).









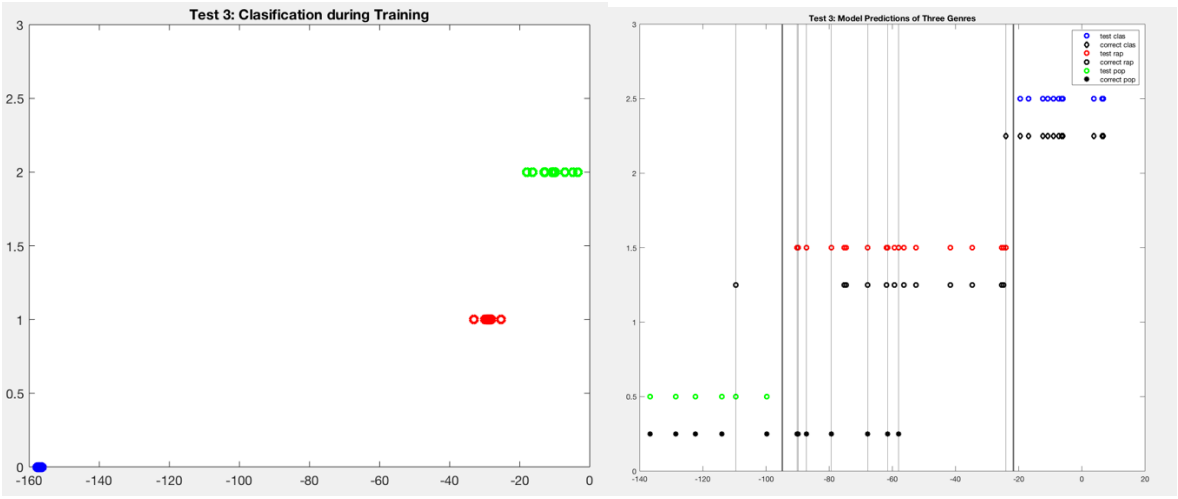


Figure 1: Collection of figures of the different tests and how well they predicted. Note that the thicker vertical lines indicate the threshold values based on the figure on the left for each test. The thinner vertical lines help indicate where the model predicted incorrectly. For test 2, the model does not predict well at all, which is why there were no "correct bts" values.

The results presented describe how well the model predicts, with the figures on the left indicating how well the model separates the different categories during training, and the figure on the right indicating how well the model does at predicting which category each song sample belongs in. For Test 1, the model predicts 31 correct and 5 incorrectly. For Test 2, the model predicts 6 correct and 9 incorrect (however it also predicts everything is under one artist, so it is not accurate at all). For Test 3, the model predicts 27 correct and 9 incorrectly.

Section V – Summary and Conclusions

Through this model, the utility of supervised machine learning using principal component analysis and linear discriminant analysis was determined. For cases in which the data is quite different, the analysis methods are able to separate the categories of music with relatively high accuracy. However, with test 2, the model fails to separate the categories of music, likely because the character of all the music is so similar. The model was able to separate the training data well, likely because the model was overfit since the first 50 modes were used, so this analysis can indicate the issues with overfitting. One of the major improvements to the quality of the result can be if the there was more training data available. Especially for some pop artists, their music can vary quite widely so they can cater to many people of the general population. More data to capture differences in music styles even within artists or within genres is necessary for this algorithm to become increasingly more successful.

Appendix A

- `[y, Fs] = audioread()`
  - This function was used to read in the audio files, giving the sample rate (44100 in each case) and the value at each sample
- `fft()`
  - Takes the Fast Fourier transform in 1D space. Used to create the Gabor transform to determine the frequency character of the music signals
- `fftshift()`
  - Shifts the values given by the fft( ) function so that they match those regularly used, because to create the spectrogram, frequencies need to start with the smallest (most negative to most positive, which is how plots are generally constructed)
- `[U,S,V] = svd(X, 'econ')`
  - This function performs the reduced SVD on a matrix, which is used to determine the principal components of the tests
- `[V2,D] = eig()`
  - This function was used to take the general eigenvalue problem which was used for LDA to determine best line to project on
- `[~,ind] = max()`
  - This function was used to find largest value of above eigenvalue problem because this is the location (ind) of the eigenvector that has the best projection