Song Genre Classification

Course Project Proposal Report

COMP 6630 Machine Learning
Dr. Shubhra Kanti Karmaker
November 10, 2022

Emma Ingram (Coordinator)
eci0004@auburn.edu

Sean O'Connor
sfo0002@auburn.edu

For our course project, we will explore the problem of classifying songs into their respective genres. Manual classification and sorting of music into genres is time consuming and tedious for our application domain of music listeners. Our desired goal is to eliminate this manual classification. By using a multi-layer perceptron we can create a solution that does this for them. Our program will take a .wav audio as input (along with its accommodating features) and output what genre the song belongs to.

We can apply a multi-layer perceptron to this problem because song genres are not linearly separable. By fine-tuning many hyper-parameters, such as the number of hidden layers (typically between the number of inputs and outputs), activation function (start with sigmoid function), weights (initialize randomly), number of epochs, momentum, batch size, and learning rate, our classifier will be able to predict the genre song belongs to. We will also use backpropagation to correct errors in our algorithm. If the classifier guesses incorrectly, we will update the weights accordingly to improve the accuracy of future classifications.

There are a few baseline models that we can take into consideration when analyzing the performance of our multi-layer perceptron approach. First, we could utilize a convolutional neural network as a baseline model. This model is typically applied to analyze visual imagery by using regularized versions of multilayer perceptrons. The difference is how this model implements regularization. To regularize, this neural network assembles patterns of increasing intricacy by taking smaller and smaller patterns collected in their filters. For music genre classification, this model would be fed in images of the spectral and temporal properties of audio signals. They enter into convolution layers, flattened into dense layers, and use a dropout layer to prevent overfitting. Another baseline model is the K-Nearest Neighbor technique. This is a supervised machine learning algorithm that looks at similar songs and assumes they belong to the same category, or genre, as the already labeled song. This approach seems to yield the best results compared to other techniques applied to song genre classification.

When it comes to the users of our classifiers, there are a variety of entities who could utilize our generated solution; however, we are going to focus on music companies. An example of some music companies who could find use of our classifier are Spotify, Apple Music, and Pandora. In terms of importance, our classifier would lead to a better population of suggested music, automate genre-specific playlist creation, and even be reimplemented to further classify genres into subgenres.

The data that we will use comes from the GTZAN dataset, which is the most-used public dataset for evaluation in machine listening research for music genre recognition. It is a collection of 1,000 30-second songs split into 10 genres - blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock. It also contains 2 .csv files with features of the audio files. One of the .csv files contains a mean and variance computed over multiple features for each song. The other .csv file has the same structure but the songs were split into 3 second audio files, increasing the amount of data by 10. We also plan to analyze a variety of features that can be extracted from the songs, including beats per minute, loudness, energy, speechiness, accousticness, valence, danceability, and discrete wavelet transform. We plan to use this dataset as-is for our project. We will split it into training and testing data and depending on how this goes, we might end up splitting the songs into smaller audio files to increase our amount of data

By creating this classifier, there are a few potential challenges we could experience. Our dataset is composed of .wav files, which could produce unforeseen obstacles since we are used to handling text files. When classifying the music, regional differences of music style could lead to improper classification. For example, a song categorized as Pop in the United States could be viewed as a Hip Hop song in another country such as Japan. In addition, certain songs could overlap into multiple genres, which could make classification extremely difficult. Ultimately, genres do not fit into one box and our classifier aims to produce a single genre for each song. These are the potential challenges we could experience; however, these are likely not the only challenges we will encounter.

To build our project, we will use python as well as the common libraries numpy, matplotlib, and potentially more. We think the assignments in this course have given us enough experience and familiarity with these tools to complete this project. Our procedure to create the classifier will be to normalize the data, create a training set, create the neural network, train the neural network, test the network, and fine-tune the hyper-parameters. This will be adjusted as needed.

We plan to show the usefulness of our software through a demo of the classifier in real time. We would like to walk through a user who wants to create a playlist given their current library of music. For example, if we input the user's music library containing songs they like, our classifier will then classify those songs into their respective library and then create a playlist of new songs our classifier suggests the user might like. The idea that our program is useful comes from the classifier being able to analyze songs much more quickly than a human, so we can gauge usefulness by comparing the time it takes to manually create a playlist of new songs with the time it takes our classifier to create one.

Below is a proposed timeline for the remainder of our project:

- Nov 10 - Turn in project proposal report
- Nov 14 - Finish loading in data and initializing program
- Nov 17 - Finish coding portion
- Dec 1 - Upload final presentation video
- Dec 6 - Complete final project report