Deep Spotify

Gaurav Lahiry, Abhy Vytheeswaran, Ronak Yarapatineni

Goals

The goal of our project was to classify a song's genre. Our motivations behind this were to improve upon the Spotify Developer API's currently available features, which does not provide genres specifically for songs (only for artists), and to see how well we could build a song recommendation system using our knowledge from this class.

Approach

We decided to approach this problem by training a neural network on a dataset of top songs from various genres and their key audio features. We decided a neural network would be a good method because of our ability to obtain a large database of songs and our goal to potentially classify songs as multiple labels of genres. Through this, we could easily build an array based binary labelling system, using discrete keyword matching to particular self-defined genres.

Other approaches we considered were reinforcement (Q) learning and Naive Bayes classification. Ultimately, we decided it would be a good idea to use the labels we had as strong determiners for correctness during training, and we also felt that the features were sometimes correlated within a particular genre and weren't as independent as they needed to be, and so we did not pursue these alternatives. We also had more experience with neural networks and felt as though it would be the most powerful and efficient way to classify our data.

Challenges

The major challenges we experienced were starting off with a suboptimal dataset given our goals, and attempting to classify songs over a large amount of genres. We started off using the Million Song Dataset which was started as a collaborative project between Echo Nest and LabROSA. Although comprehensive, this dataset encompassed a very diverse array of songs, most of which fit into obscure genres. Furthermore, many of their audio features did not match up to audio features of traditional songs we'd expect for each genre.

Thus, we pivoted to using a dataset of songs from Spotify's featured playlists. This way, we were able to get an equal amount of songs from a number of popular genres. The second challenge we faced was attempting to classify our songs into a large number of genre labels. We were trying to distinguish between different subgenres that turned out to be too similar (for example, we had different labels for pop, EDM, house, dance but the latter three often fit under the pop umbrella). Thus, we decided to reduce the number of genre labels we were classifying the songs into from about 14 to 8 by combining similar subgenres into one larger genre.

Our third major issue resulted from our resolutions of the first 2. The choice to use more contemporary versions of a variety of songs meant that some of the input features, across

genres, were more similar than we were hoping for. That is to say, the features provided by the Spotify API were more closely aligned even across genres, even though we knew the sound was actually distinctive. The problem we encountered here was an introduced bias towards our first label, pop, which was devolving into a “popular” categorization rather than true “pop”. After noticing this, we looked at many outputs of our classification, and realized that true pop songs had very low probabilities for all other genres, whereas something being incorrectly classified as pop would have a decent score in their true category. Thus, we adjusted this bias and classified such songs as this other category, if they were over a certain threshold.

Implementation

We trained our neural network on 12,000 songs from Spotify’s featured playlists. The inputs’ attributes were 10 audio features that we obtained from the Spotify API, which included instrumentalness, danceability, loudness, energy, liveness, tempo, acousticness, speechiness, valence, and mode. We built our neural network with a learning rate of 0.3, one hidden layer comprised of 30 hidden units, and outputs of 8-wide binary vectors that represented genre labels, including pop, hip-hop, indie-alternative, metal-rock, jazz, classical, R&B, and country. Additionally when training our net, we were sure to normalize the values that Spotify gave for a song’s audio features, decided to use the sigmoid activation function, and used MSE for backpropagation.

We tested our neural network on a different set of 12,000 songs from Spotify’s featured playlists. As we stated earlier, when testing, we found that the incorrect predictions were often mislabeled pop, but that the genre with the second highest probability was the correct genre. Thus, we decided to accept these if the genre with the second highest probability was above a threshold of 0.5. This is also how we outputted the predicted genre(s) for a given song: we output two genres for a song if the second highest was greater than our threshold, and output one otherwise. Also, since so many songs classified into pop (the audio features are very similar to that of rock and hip-hop), we outputted only genre with second highest probability if the first was pop *and* the second was was above our threshold.

Finally, when predicting a song genre, we used the set of weights that resulted in minimum error which we had saved, and normalized its audio feature values with respect to the range of values we found for our training data which we had saved. We built a simple user interface for this as a local Flask web app, which predicted an inputted song’s genre(s) and also suggested five songs based on these genres using the Spotify API.

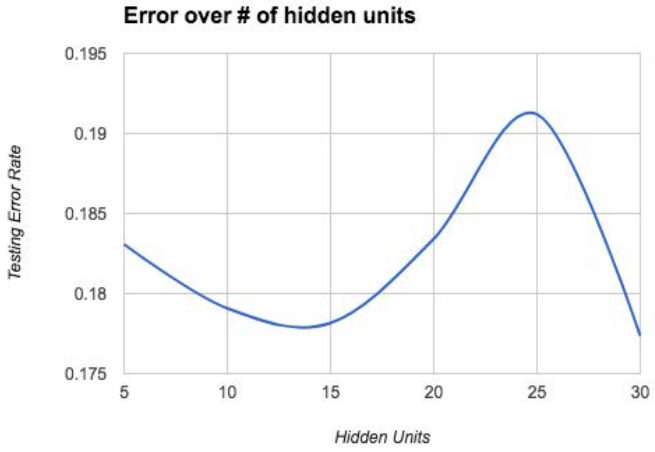
Results

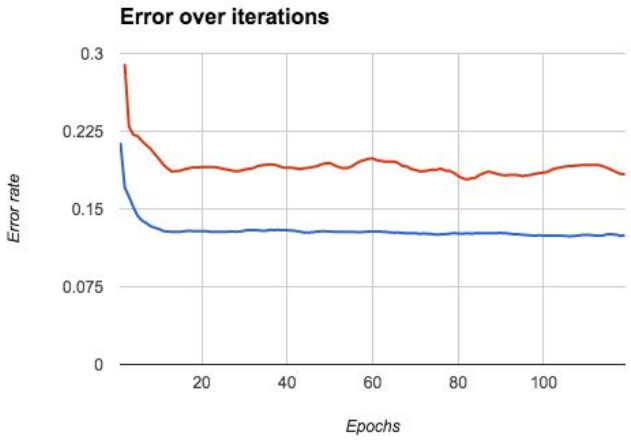
We nearly accomplished our goal of classifying a song’s genre using the Spotify API. Although we were able to classify many songs with high accuracy, this was more so for some genres than others. Furthermore, we weren’t able to output genres of the variety that we were initially aiming for. For example, although we classified most pop, hip-hop, metal-rock, and indie-alternative songs correctly, most classical and jazz songs were classified as metal-rock. These types of

errors were likely due to lack of distinctiveness in their features (such as instrumentalness) and a smaller amount of data for particular genres.

Error Analysis

Our minimum training error rate was 0.124 (88%), and testing error rate was 0.177 (82%). The main source of our error was largely due to the lack of easily available data for the types of songs we wanted to be dealing with. If this data was available, we could have spent more time processing our training data, by adding an equal amount of songs from all genres. Doing this would have especially improved the accuracy for songs with lower quantities of data, which caused us the most issues. Furthermore, our errors in general were probably also due to the fact that Spotify provides genres according to the artist, not the song.





The graph on the left shows our analysis of choosing the optimal number of hidden units. We found that increasing the number of hidden units up to 30 provides the best error values for both our training and testing sets. Increasing past this causes the error rate to shoot back up.

The graph on the right shows our errors over 120 iterations, using 30 hidden units. It shows a typical decrease in error rate as we increase the number of epochs, and does a good job of not getting stuck in local minimums with a learning rate of 0.3. It also avoids overfitting, as we see the testing error (red) decrease in the same fashion as the training error (blue).

Moving Forward

In order to improve upon our implementation, we first should do what we can to minimize error. This includes spending more time processing training data, adding an equal amount of songs from all genres, and improving our labels. After we do this, we can more easily classify our data into multiple genres and more specific subgenres, as the quality of our training data is higher. As a result, we can recommend more similar songs that have greater relevance to the input song. Additionally, instead of relying on the Spotify API's audio features, we could instead use a song's .wav file by using Fourier's transform to convert the audio data to the frequency domain and obtain

the evolution of all the frequencies of the song over time. This could potentially give us better input features to train on.