Using Song Lyrics and Frequency to Predict Genre

Jayen Ram
Stanford University
`jjram@stanford.edu`

Daniel Salz
Stanford University
`dasalz@stanford.edu`

1 Introduction

In the age of music streaming and crowd sourced artistry, there are thousands of songs and videos being uploaded to services like YouTube and Soundcloud daily. On the other side, millions of consumers are scavenging these websites for new music; currently, many social media services give suggestions to users by genre and type, and uploaded music from the community is often unclassified and cluttered in the amalgamation of media. A potential solution would be a machine learning algorithm that automatically categorizes these songs. By implementing an automatic classifier, music services can give artists the option to publicize large sets of their own music without having to manually select what each song genre falls under. In some cases a song can also be incorrectly classified by the artist, a service like this could correct this problem and avoid delays and complications in music deployment.

2 Related Works

We found that holistically there were two sides of the music genre classification problem tackled by researchers; one side looked at lyrics of songs and constructed a model that would translate this into a genre, and one side looked at either the MIDI break down or the Fourier transform vector of songs to create predictions.

For the papers that deal with the actual frequency vectors or sound properties, we found a few ensembles of algorithms applied to the problem. Sanden *et al.* uses support vector machines, k-nearest neighbors, and decision tree to make multi-class predictions that uses the top-k class labels based on the top n scores allowing them to select the model that predicts the best for a given input [1]. These researchers, and most others, used MFCC, ZCR, Spectral Centroid, Rolloff, Spectral Flux, and Chroma as their feature vector. I found that their strength was the implementation of numerous techniques and several datasets to find the best for recall, precision, etc., although the paper maintained the same set of features. Gouyon *et al.* used rhythmic descriptors instead of the more standard audio features used by other papers [2]. They classified sub-genres of ballroom music using features like mean tempo, max tempo, tick, salient periodicity, PH centroid that had to do with music structure instead of wave analysis. Although the amount of data used by the paper seemed somewhat low, it seemed that their model still found that rhythm was the most important aspect in classification. We also explored the dissertation from Tzanetakis *et al.* who used a similar weighted ensemble of algorithms, that seems very successful in getting high classification results [3]. A common theme was that the highest accuracy many achieved was around 50% because of the multi-class classification.

When reviewing lyrics classification, we found Sadovsky *et al.* who use a bag of words technique to select words to classify using an SVM technique; they use sentence structure to create a bag of features including part of speech, repetition, and certain words. Their model was quite effective in differentiating between 3 genres, even though their feature vector mostly was impacted by the words themselves [4]. Tsaptsinos uses several genres (over 117) and uses a hierarchical attention network; this reduces the need to evaluate features against each other. This model uses sentence order to create a probability of each lyric combo being part of a certain genre [5].

Submitted as Final CS 229 Project

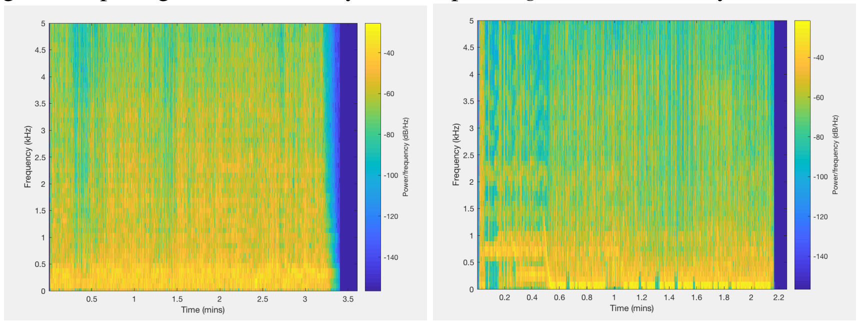
3 Dataset and Features

3.1 Lyrics Classification

We decided that For our lyrics classification model that used lyrics as input for classification we used a Kaggle dataset that contained an artist, a list of songs for that artist, and the lyrics for each song. In total the data set contained 55,000 different songs. The labels for our data were created based on the artist of the song. We went through all the artists in our data set and assigned them a respective genre based on either our own knowledge or the top result given by Wikipedia. We then determined the artists that had music that would fit in a genre within country, rap, and rock (the genres we would attempt to classify), and divided the dataset accordingly. In the dataset, there were 4701 rap songs, 1446 country songs, and 3775 rock songs from the artists we determined were fit to be added to our dataset. After selecting the songs, we determined a group of words that would be indicative of the various genres by first finding 500 words that were found in the 1000 occurence to 3000 occurrencce range in the three categorie

3.2 FFT Classification

Figure 1: Spectogram of *D Rose* by Lil Pump vs *Big Green Tractor* by Jason Aldean



We decided to use FFT as our feature vector for the Neural Net because different genres of music utilize different sounds in their songs. For example rap songs contain significantly more bass as they use 808s and beats primarily as their sound source. These sounds are represented uniquely by the frequency and amplitude qualities in the FFT vector of a song. A Neural Net is used to weight these frequencies in order to classify them to a specific genre. The figure above displays the frequency data two songs from different genres contain. To obtain the data for our Neural Net we wrote a python script that downloaded the audio of the top Youtube search result of the song in wav format. By downloading all the audio files in wav format we could ensure that they were sampled at the same audio frequency. The song is then processed through MATLAB using the *audioread()* and *fft()* functions to obtain an FFT vector of the song. We then only took the first 2,000,000 buckets and sampled every 8th value. This flattened vector is a 1x250,000 feature vector for the Neural Net. For binary classification of rap vs non-rap, our data set contained 600 rap songs and 6600 non-rap. The 3 class NN used the same songs as the 3 class SVM.

4 Methods

4.1 Naive Bayes

Our first implementation to solve the lyrics classification problem was to implement a binary and multinomial Naive Bayes algorithm that took in an input x with 539 features (the 539 words selected by our word extractor algorithm). The output vector was a label 0, 1, 2 represention [rock, rap, country].

Our Naive Bayes implementation used a Laplace smoothing of 2.0, which we tuned with the dev set; the dev set accuracies can be seen below for the various λ tested for the Naive Bayes:
We implemented Naive Bayes for the lyrics classification because this problem is much like the spam detection problem posed in a CS229 problem set; certain words will be likely to influence the likelihood of a song being of a particular genre. One can imagine some words that would likely make a rap song a rap song, etc. For our probabilities we come up with the following equations:

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \& y^{(i)} = 0\} + 2}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} + 6}$$
$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \& y^{(i)} = 1\} + 2}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} + 6}$$
$$\phi_{j|y=2} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \& y^{(i)} = 2\} + 2}{\sum_{i=1}^{m} 1\{y^{(i)} = 2\} + 6}$$

4.2 Support Vector Machine

From literature review, we found across the board that text classification done with Naive Bayes is generally improved with a SVM algorithm. Upon research on text classification methods, we settled on using an RBF kernel instead of a linear or polynomial kernel because this problem will not likely be solved with a linear classifier. Our SVM uses the following kernel function:

$$K(x, x^{'}) = exp(-\frac{||x - x^{'}||^2}{2\sigma^2})$$

In our implementation of the SVM, we also included a regularization term that is a part of the sklearn library that is defined as such:

$$C \sum_{i=1,n} \mathcal{L}(f(x_i), y_i) + \Omega(w)$$

This regularization term allowed our train and test/dev errors to be relatively close and avoid the overfitting that sometimes comes with a SVM implementation.

4.3 Neural Network

The input data x is: $(x_{(i)})_{i=1:n}$ where each index is one of the sampled buckets of the Fast Fourier Transform vector of the song we want to input.

The number of input nodes to the graph is therefore 250,000. The output data is a one-hot vector to indicate which genre the data was classified to. The size of the output was 2 for binary classification (rap vs non-rap) and 3 for for multiple classifications (rock, country, and rap).

Our Neural Net used two hidden layers with 100 neurons and no activation function g(z) as the sigmoid function. $\frac{e^x}{e^x+1}$The output layer used softmax(z) for its activation function: $\frac{exp(z)}{\sum_k exp(z_k)}$ We use softmax at the output layer because for classification we want the Neural Net's output to only have a single high bit in the output, representing the classification as the one-hot vector described above.

Our cost function was the cross entropy loss of the softmax output, shown by: $CE(y, \hat{y}) = -\sum_{k=1}^{K} y_k log \hat{y}_k$
Where $\hat{y} \in \mathbb{R}^K$ is the vector of softmax outputs for a single example x, and $y \in \mathbb{R}^K$ is the actual label for the input x. Our Neural Net uses mini-batch gradient descent as its a good balance between stochastic gradient descent and batch gradient descent, which would be too computationally expensive especially considering the large size of the input vector. With mini-batch our cost function becomes: $\frac{1}{B} \sum_{i=1}^{B} CE(y^{(i)}, \hat{y}^{(i)})$

On each iteration of training the Neural Net uses backpropagation to calculate the gradients. The gradients are calculated as such:

$$\text{Output Layer : } \delta^{[N]} = \nabla_{z^{[N]}} \mathcal{L}$$

For the hidden layers the gradients for the weights and biases are computed as:

$$\delta^{[\ell]} = (W^{[\ell+1]T} \delta^{[\ell+1]}) \circ g^{'}(z^{[\ell]}); \nabla_{W^{[\ell]}} J(W, b) = \delta^{[\ell]} a^{[\ell-1]T}; \nabla_{b^{[\ell]}} J(W, b) = \delta^{[\ell]}$$

The parameters of the Neural Net are then updated accordingly to the gradients computed in back-propagation, the cost function from forward propagation, and current learning rate.

5 Experiments/Results/Discussion

5.1 Lyrics Classification

The first iteration of our implementation involved a binary classification for classifying songs as rap or not-rap. The dataset involved 4701 rap songs and 10000 random non-rap songs. We implemented a simple Linear Regression, an SVM, and a Naive Bayes model for this initial problem and achieved the below accuracies:

	Lin. Reg.	Naive Bayes	SVM
Train Accuracy	89.7%	92.3%	94.4%
Test Accuracy	88.5%	91.6%	93.2%

As shown above, the lyric classification problem was best solved by the SVM with RBF kernel, and in all cases the trained model fitted well with the test set. When analyzing the top 5 words in rap songs that led to its classification, we found that they were extremly indicative of the genre (we will leave them out of the report for decency), and was interesting in showing how these songs had a certain subset of words that were unique.
When analyzing the multiclass versions of these algorithms, we compared rap, country, and rock songs with the following confusion matrix for our SVM classification:

	Predicted Rock	Predicted: Rap	Predicted: Country	
Actual: Rock	341	10	65	416
Actual: Rap	13	110	21	155
Actual: Country	156	6	189	351
	510	126	275	

We can see from the matrix, the classifier mixed up country and rock more than with rap, which matches intuition since words like "bottle," "wild," and "whiskey" were found mostly in either of the two but were featured in several songs of the other genre. Unsurprisingly rap classification trounced that of country and rock; upon further inspection, it seemed that words like "y'all" and others that were featured in country songs were also found pervasively in rock songs.

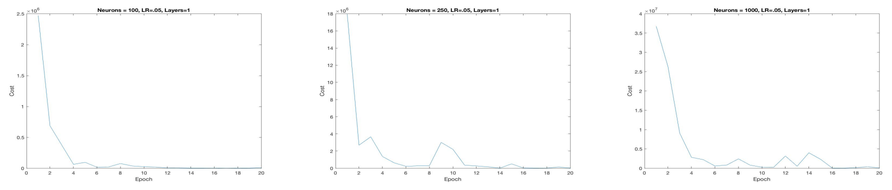
5.2 Neural Network and Frequency Classification

In our initial experiments of classifying genre based on Frequency we focused on binary classification, more specifically rap vs non-rap. For these tests our dataset was half rap and half nonrap, the batch size was 20, the learning rate was .05, and had one hidden layer of 1000 neurons. The results of these experiments were underwhelming as our test accuracy varied between 69-76%Ẇe decided to re-evaluate our data because the model looked it was overfitting.Total number of songs changed from 1000 to 7200 (600 rap/6600 nonrap).We then re-evaluated our labels to be more accurate to the songs. With our new dataset we were able to achieve 88% accuracy, a significant improvement.

We now wanted to get the best parameters for the mode. First we tuned the number of Neurons in the single hidden layer, we started with 1000 neurons because our input vector was very large, however we were curious if a lower number was still suitable.

	100 Neurons	250 Neurons	1000 Neurons
Train Accuracy	91.4%	91.1%	91.6%
Dev Accuracy	88.7%	88.5%	88.2%

Figure 2: Cost vs Epoch for 100 vs 250 vs 1000 Neurons



The accuracy was pretty much the same regardless of the number of neurons, however the higher number of neurons seemed to make the cost more volatile across iterations. We decided to stick with

100 neurons in the hidden layer from this point onward because its less computationally expensive and its cost iteration was more reasonable.

We then experimented with the number of hidden layers. Our initial tests only used a single layer so we tested adding 1 and 2 more layers.

	1 Hidden Layer	2 Hidden Layers	3 Hidden Layers
Train Accuracy	91.4%	93.6%	89.5%
Dev Accuracy	88.7%	91.4%	87.4%

Adding two more layers was detrimental to the model, but adding a single one improved our accuracy from both training error and dev set error. We changed our model to stick with two hidden layers for the later tests.

The last model parameter we experimented with was the learning rate. Changing the learning rate is where we saw the best increase in accuracy.

	LR=.05	LR=.01	LR=.005	LR=.0025	LR=.001
Train Accuracy	93.6%	96.5%	97.2%	97.5%	93.7%
Dev Accuracy	91.4%	93.3%	95.6%	94.8%	91.0%

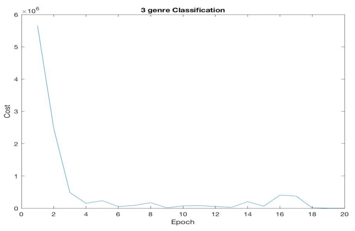
Using the results of cross validation on the learning rate we concluded our parameters for the model: Learning Rate = .005, 100 Neurons in Hidden Layer, Two hidden layers, and Batch Size of 20. When retraining on the entire data set of 7200 songs, our test accuracy was 95.5%.

N = 200	Predicted Rap	Predicted: Non Rap	
Actual: Rap	51	6	57
Actual: Non Rap	3	140	143
	54	146	

We also attempted making our input vector smaller by replacing every 100 values by their mean and turning our 250,000 input into 2,500 features. When using this new input feature though our accuracy decreased to 87% and we decided to stay with the original input vector size.

5.2.1 Multiple Genres

Once we were confident in our model for binary classification we attempted to expand it to multiple genres. Using the same model parameters as above, we tested the Neural Net on 3 classes, this data contained 4701 rap songs, 1446 country songs, and 3775 rock songs. We achieved 69.5% accuracy on this model.



N = 200	Predicted Rap	Predicted: Rock	Predicted: Country	
Actual: Rap	72	11	3	86
Actual: Rock	16	44	21	81
Actual: Country	8	16	9	33
	96	71	33	

6 Conclusion/Future Step

Overall, we found that with both lyrics and sound classification, binary was much easier than multi-class especially for rap music; rap music, as shown by our results, is quite discrete as a music genre and has little overlap with other genres holistically. In the future, we would like to address the multi-class problem with an ensemble of Machine Learning methods that could combine to create a confidence vector for each input, such as using our SVM and NN models together. This would expand our findings to be used for a streaming service that can input several different genres and have some interesting results. In addition, we would like to experiment with different types of neural networks (recurrent and convolutional) to see if this will expose different implicit features in the dataset.

7 References

[1] C. Sanden and J. Zhang. Enhancing multi-label music genre classification through ensemble techniques. In Proc. of ACM SIGIR, 2011

[2]F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, “An experimental comparison of audio tempo induction algorithms,” IEEE Transactions on Audio, Speech and Language Processing, vol. 14, no. 5, pp. 1832– 1844, 2006.

[3] G. Tzanetakis and P. Cook, “Music genre classification of audio signals,” IEEE Trans. Speech and Audio Process., vol. 10, no. 5, pp. 293–302, Jul. 2002.

[4] Sadovsky, A., and Chen, X. (2006). Song genre and artist classification via supervised learning from lyrics. Student Report Stanford University.

8 Contributions

Jayen Ram - Headed the parsing of Lyrics Classification dataset, token extraction, development of Naive Bayes, and SVM algorithms, assisted with neural net progress and music download

Daniel Salz - Led the construction of neural network, downloading and parsing of music dataset, assisted with lyrics classification aspects and data download.