Musical Genre Classification

Bibit Bianchini
Mechanical Engineering
Stanford University
bibit@stanford.edu

Thomas Froger Silva
Mechanical Engineering
Stanford University
tfrogers@stanford.edu

***Abstract*—This paper presents different ways to extract features from audio snippets and different learning algorithms that allow to classify songs by there genre. The features we extracted were Zero Crossings, Spectral Centroid, Spectral Roll-off, Mel-Frequency Cepstral Coefficients, and Chroma Frequencies. For the learning algorithms, we used support vector machines using linear approach with batch gradient descent and stochastic gradient descent. Our dataset had 1000 30-second excerpts of songs classified in 10 different genres. Despite the small dataset, we managed to get an overall 26% accuracy on our test set and up to 80% accuracy for recognizing classical music in particular.**

I. Introduction

Identifying musical genres is a task useful for a multitude of different applications in the music industry, from new music discovery algorithms to simply setting a new (or recently discovered song) into its right genre. It is common for discovery algorithms to rely on ratings of songs in order to recommend them to people whose tastes align with those who rate the song highly. However, this solution relies on others' data in order to know anything about the song. If we can create an algorithm that classifies music into different categories, then we would be able to recommend brand new songs without data about other users' opinions about the song and see if the user like a specific genre. For other apps like Apple music and Spotify, they would be able to quickly sort songs by genre when they add them to their vast libraries.

The input for our algorithm is a 30 second snippet of a song. We converted the audio to into a number of different features: zero crossings, spectral centroid, spectral roll-off, mel-frequency cepstral coefficients, and chroma frequencies. We then used SVM and SGD machine learning algorithms to output a predicted genre of a song from an option of 10 genres: reggae, disco, classical metal, country, hiphop, jazz, pop, rock, and blues.

II. Related Work

There was a well-known paper by Tzanetakis et. al about identifying music genres from the GTZAN dataset.[1] This work generated five pitch-related features and five rhythm-related features from each audio signal. Generating features from the audio files turned out to be the most time-consuming portion of our machine learning project, thus, reducing the number of features would greatly speed up the rate at which deploying a genre-identification algorithm could happen. For this drawback, we drew inspiration from this work, though we intend to see if we could use a smaller number of features to do the predictions.

We took further inspiration from Pandey, who worked on the same problem of identifying music genre from audio snippets but with different features than the Tzanetakis et. al paper.[2] Pandey only generated 5 types of features, half of that of Tzanetakis et. al. Pandey then generated colorized plots from these values and used image processing machine learning algorithms in order to make the predictions. This image generation step additionally adds time to the data processing step, despite Pandey's reduction of feature generation. Thus, we intend to use the smaller number of features similar to those Pandey generated but with numerical machine learning algorithms such as logistic regression, support vector machines, etc.

An additional paper we found looked into detail about how to pick features for an automatic music genre classification algorithm.[5] A distinguishing aspect of this paper was the fact that the data was extracted from the beginning, middle, and end of each song. This could help with extracting characteristic features of songs especially in some genre of songs where the audio could change quite a bit depending on where the excerpt is from. Similar to our paper, the authors used SVM classification to for the classifier algorithm. They also used other algorithms such as Naive-Bayes, decision trees, and multi-layer perceptron Neural Nets. Their test dataset was also much larger than ours (3,227 songs). Having more data could have helped our analysis. One drawback of this paper is that they did find some features to be more discriminative than others, yet they did not remove them and run the learning algorithms again.

III. Dataset

We are working with the same GTZAN genre collection dataset that was used by Tzanetakis et al.[4] This dataset comes with 1000 pre-trimmed audio files of 30 second snippets from 10 different labeled genres -- 100 audio files per genre. We did a 60% - 20% - 20% split for training, cross-validation, and test sets, ensuring that the proportion of each genre was consistent.

Prior to doing any training, we performed data processing on the audio files in order to extract features. We used five wave file analysis techniques on the audio data:

- Zero Crossings
- Spectral Centroid
- Spectral Roll-off
- Mel-Frequency Cepstral Coefficients
- Chroma Frequencies

The Zero Crossings number quantifies how many times the audio waveform crosses the decibels=0 line over the 30 second song snippet. The Spectral Centroid analysis produces a vector of quantities of the average concentration of frequencies for different timestamps within the audio waveform. Similarly, the Spectral Roll-off analysis produces a vector of quantities of the "roll-off frequency" of a waveform for different timestamps within the audio waveform. The Mel-Frequency Cepstral Coefficients (or MFCCs) collectively make up a representation of the short-term power spectrum of a sound for different timestamps within the audio waveform. This MFCC method generates a matrix of values, where one dimension corresponds to 20 power groupings and the other dimension corresponds to the time within the audio snippet. The Chroma Frequencies are similar to the MFCCs in that they project the audio waveform into 12 bins that correspond to the 12 tones within the standard musical octave (a scale) for different timestamps.

We generated features using each of these 5 methods for all of our audio files. We were able to generate images such as the following for reggae, blues, and classical examples:

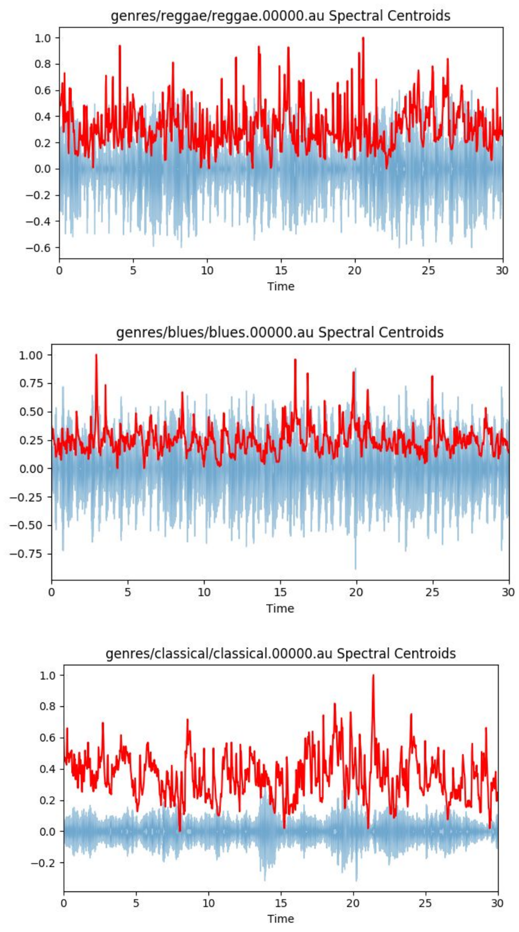


Figure 1: Spectral Centroids plotted over each original audio files' waveform.

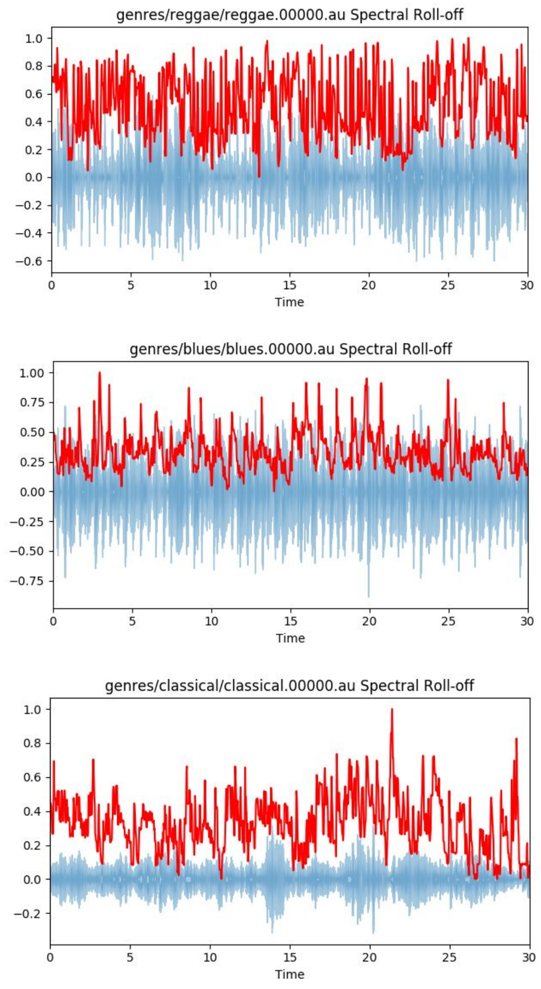


Figure 2: Spectral Roll-offs plotted over each original audio files' waveform.

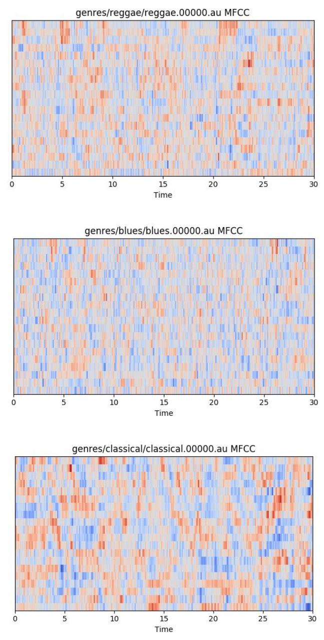


Figure 3: MFCCs.

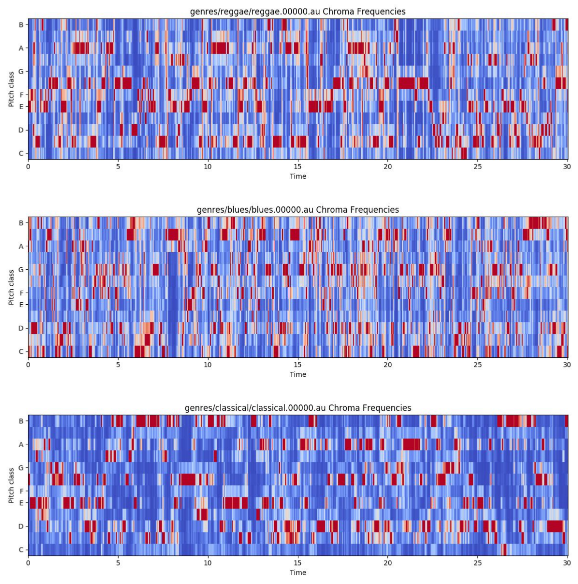


Figure 4: Chroma Frequencies plotted.

Once we verified that the audio data analysis functions were working, we worked on automating this process for large numbers of samples at a time. From audio inputs, we were able to generate csv files containing any subset of the generated quantified values obtained via the above described methods.

We tried these techniques for 2 different sampling rates of the audio files: 22 kHz (the original sampling rate of the provided audio files from the dataset), and 11 kHz. Thus, the following yields the number of features we had for each example:

Sampling Rate	**22 kHz**	**11 kHz**
Zero Crossings	1	1
Spectral Centroids	1,285	640
Spectral Roll-off	1,285	640
MFCC	25,700	12,800
Chroma Frequencies	15,420	7,440
Total Features	43,691	21,761

Using these new values as features, we began implementing a machine learning algorithm to work on classifying the data.

IV. Methods

We used support vector machines (SVM) with two different gradient descent methods. For the first algorithm we used a linear support vector machine and one-vs-the-rest scheme using batch gradient descent and for the other we used SMV with stochastic gradient descent (SGD). Support vector machines is a group of supervised learning methods used for classification and regression analysis.

When using linear SVM, our goal is to find the maximum-hyperplane that separates a genre of songs from the other genre. To predict the genre of a new sample we calculate the probability of the song to be part of a genre and pick the one with the highest likelihood:

$$h_\theta^{(n)} = P(y = n \mid x;\ \theta)$$

$$prediction = max_i(h_\theta^{(i)}(x))$$

The hypothesis equation used for linear SMV is:

$$h_\theta = \frac{1}{1 + e^{-\theta^T x}}$$

The cost function used for linear SMV is:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\, log(h_\theta(x^{(i)})) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)}))$$

We then use this cost function to calculate the parameter θ that best fits the data. We use gradient descent to optimise θ:

```
Repeat {
    $\theta_j := \alpha * \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})\, x_j^{(i)}$
    (simultaneously update $\theta_j$ for all $j$ = 1=0, …, n)
}
```

We decided to use Python's SciKit Learn package. Using this package, we were able to change different parameters in the algorithm. We had more features than samples, so we decided to solve the dual optimization problem. We set a very low tolerance to insure that our algorithm would be more accurate. We iterated over multiple number of maximum iterations and tested our algorithm on a cross-validation set to find the best possible number of iterations.

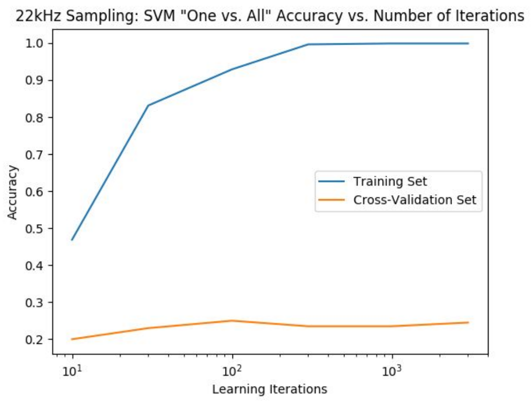
For the stochastic gradient descent approach we used the SGDCLassifier package. The main difference between these two approaches is the method used for gradient descent. For stochastic method, we approximate the gradient using a single example with the following update function:

$$\theta := \theta - \alpha * (h_\theta(x^{(i)}) - y^{(i)})\, x_j^{(i)}$$

The algorithm sweeps through the dataset, performing this update one example at a time. We also used an adaptive learning rate generated by the SciKit Learn library to ensure that the algorithm converges properly.

V. Experiments/Results/Discussion

First, for each of our models on each of our datasets, we verified that we were able to learn parameters correctly. We were looking for the training accuracy to approach 100% as we increased the number of iterations of running the learning algorithms. We were able to see this phenomenon for all 4 combinations of 22 or 11 kHz sampling for both our SVM and SGD algorithms.



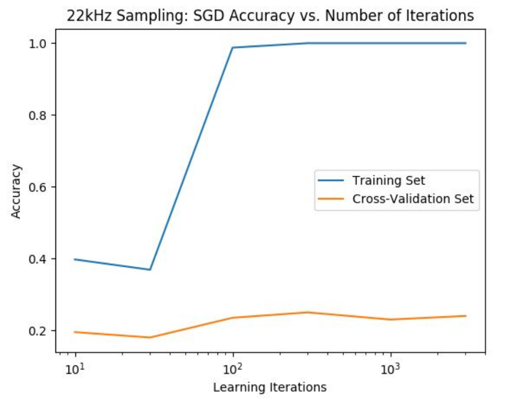
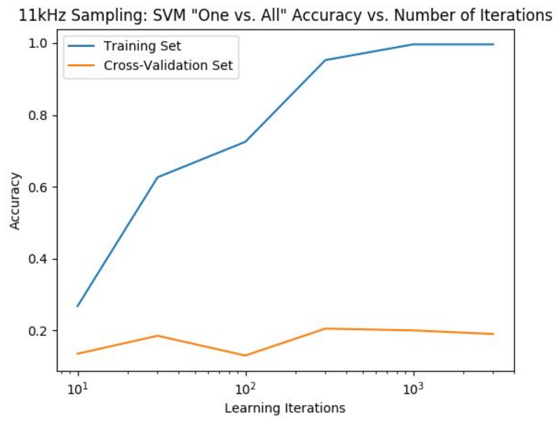


Figure 5: Learning curves for SVM and SGD algorithms for the 22 kHz sampling rate dataset.



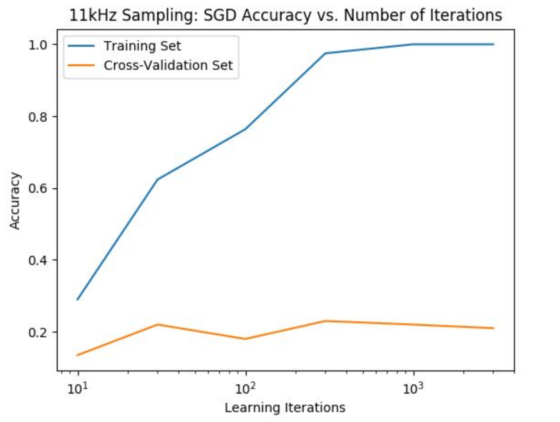


Figure 6: Learning curves for SVM and SGD algorithms for the 11 kHz sampling rate dataset.

After generating the learning curves, we determined with the cross-validation performance that 300 iterations yielded the peak accuracy for all but 1 of the four combinations (SVM on the 22 kHz sampling rate dataset did better with 100 iterations). We then used this 300 iterations parameter to run our analysis on our test sets, and we obtained the following performance:

Model / Accuracy	**Training Set**	**Cross-Validation Set**	**Test Set**
SVM @ 22 kHz sampling rate	100%	22.5%	24.8%
SGD @ 22 kHz sampling rate	100%	24.5%	26.3%

We speculate that our test set accuracy increased in comparison to our cross-validation set because of small dataset round-off errors. We hypothesize that with a larger dataset that there would be another decrease in accuracy from the cross-validation set to the test set.

We further analyzed the performance by genre on our test sets, this time for all four combinations of sampling rates and models.

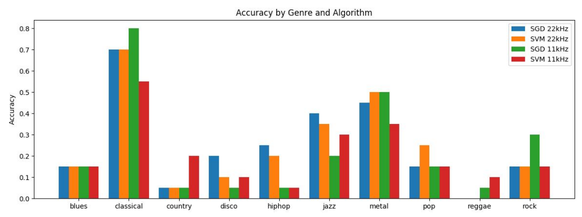


Figure 7: Accuracy within each genre for each combination of machine learning model and data sampling rate.

It is clear that classical music is the easiest for all of the models to identify from the others with an average of 70% test set accuracy for that genre. Metal and jazz were the second most distinct genres for the algorithms to identify, though not as consistently as classical. Reggae was the most difficult to identify among the other genres, as samples of this genre were most commonly mistaken for jazz or blues.

VI. Conclusion/Future Work

In terms of data processing, we could put in more time to identify the most helpful data analysis techniques. Our work relied on 5 features, though we did not test out any additional features in order to determine which worked the best. More than data processing, we believe that the accuracy of our models could go up most significantly by expanding the size of our dataset. Our training set only had 60 examples per each of the 10 genres, and we believe that a number in the hundreds or thousands could yield better results. Particularly since our number of features is so large (10s of 1000s), increasing the number of examples could open the door to more algorithm possibilities.

The two algorithms we used seemed to have very similar performances. I suspect the lack of training data lead to both algorithms having similar results. Future work could involve trying other algorithms such as neural networks that tend to help when the number of features greatly exceeds the number of training examples. Potentially also using the same algorithm with fewer features (for example taking characteristic values of features we already had such as mean, standard deviation and mean) could help with the performance of the two SVM methods.

Acknowledgments

We would like to thank the entire CS229A course staff for their help on this project.

References

[1] Tzanetakis, G., and P. Cook. “Musical Genre Classification of Audio Signals.” IEEE Transactions on Speech and Audio Processing, vol. 10, no. 5, 2002, pp. 293–302., doi:10.1109/tsa.2002.800560.

[2] Pandey, Parul. “Music Genre Classification with Python.” Towards Data Science, 12 Dec. 2018, towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8.

[3] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[4] Leben, Jakob. “Data Sets.” Music Analysis, Retrieval, and Synthesis for Audio Signals, 2015, marsyas.info/downloads/datasets.html

[5] C. N. Silla Jr., A. L. Koerich and C. A. A. Kaestner, "Feature Selection in Automatic Music Genre Classification," 2008 Tenth IEEE International Symposium on Multimedia, Berkeley, CA, 2008, pp. 39-44. doi: 10.1109/ISM.2008.54