Music Genre Classification

Ryan Hoang and Marwan Ali

CS 484 - Data Mining
December 11, 2018

Abstract

Our final project focuses on music genre classification. We used two methods. The first was based on using extracted audio feature data and the second was based on using lyric data. Both the audio feature and lyric datasets are subsets of the Million Song Dataset. The audio dataset contains 59600 tracks and the lyric dataset contains 34953 tracks. One of the biggest challenges we encountered was how unbalanced both datasets are. We tried countering this by giving minority classes higher misclassification penalties or resampling the original data to get a better balance between the genres. We experimented with many different models on both sets of data. We found that for the audio data, random forest classification performed the best getting an F1-Score of 58%. For the lyric data, linear SVMs performed the best scoring a 72% F1-Score.

1 Introduction

Music is deeply subjective and we all relate to it in different ways. Music classification and recommendation is a challenge that big consumer companies are facing and will continue to face as more music is released and more users sign up for these products. Music data is being created at a rapid pace and it will be a challenge to organize and analyze it in a way that is meaningful for users and will create value for the companies. How do you create a playlist that seems like it was tailor made for each user and that you know they'll love? What features could you pick out from a song that can help determine if a user will love it or not? What level of granularity guaranties accurate classifications? Ultimately, that is what we attempted to answer while doing our project.

2 Problem Statement

Given either audio features or lyrics for a song, how effectively can we predict its genre? Does lyric data give different results than audio data?

3 Literature Review

Music Genre Classification Using Support Vector Machines

This scholarly article describes an approach to music genre classification using a multi-layered classifier which was based on multiple support vector machines. The authors of this paper used a dataset composed of audio signal data recorded from 100 different audio tracks. Each of these tracks was segmented into many different parts for a total of 200,000 data points representing music from pop, classic, rock, and jazz. Their method was to use multiple SVMs one after another in order to separate the music into different genres. The first SVM split the data into pop/classic and rock/jazz. The second SVM split the pop/classic set of tracks into pop and classic. The third SVM split the rock/jazz set into rock and jazz. They achieved accuracies of over 90% for each of the SVMs. They also compare these results with other methods used in the same fashion i.e. Neural Networks, Gaussian Mixture Model and hidden Markov Model.

Exploring Different Approaches For Music Genre Classification

In this paper the researchers used combined SVMs to classify genres. Three SVMs were trained to return whether a track fell in a 3 categories of genres instead of individual genres and the the results from all three SVMs are combined for final results. The researchers only used time and frequency feature vectors to do this. When they used 10% of the data for training, the accuracy ranged from 51.8% to 92.6%. They were able to achieve 100% accuracy when they used 80% of the data set for training. They concluded that strong results could be achieved with only 10% of the data making their approach easier to generalize.

Besides the approaches mentioned above for genre classification, there are also approaches that try classifying based on moods instead of genres. I could see have this would be useful for creating playlists that are composed of multiple genres like Spotify's study or party playlists.

4 Methods and Techniques

The original audio dataset is comprised of 59600 tracks. The first thing we had to do was break it up into training and test datasets. To do this we used the Pandas library for filtering data and creating datasets that were well proportioned for each genre. Since the audio dataset isn't balanced we decided to use 30% of the tracks per genre to create the training dataset and use the remaining 70% for the test dataset. One of the ways we tried to work around the imbalanced dataset problem was giving the classes different weights or penalties for a misclassification. For example, since hip-hop has significantly less entries than folk, we made the cost of misclassifying a hip-hop song much higher than that of a folk song. This can be tricky though. These penalties can bias the model to pay more attention to the minority classes, at the expense of the majority classes. Another work around we tried for the imbalanced data was resampling the original dataset by deleting instances of the over-represented class.We also decided to drop some of the data that was in the original dataset from the training and test datasets. The original dataset contained artist and title columns that we didn't want to include because it was too specific.

5 Discussion and Results

5.11 Audio Dataset

The audio dataset is a subset of the Million Song Dataset (MSD) that was compiled by researchers at Columbia University. They wanted to create a simplified genre dataset from the MSD for teaching purposes. They used artist tags that were included in MSD and matched them against data from musicbrainz to get genre data. From the top 50 most popular musicbrainz tags, they picked 10. The 10 they chose are classic pop and rock, folk, dance and electronica, jazz and blues, soul and reggae, punk, metal, classical, pop, and hip-hop. They excluded any artists that could be matched to multiple genres to avoid getting misleading classifications. They ended up with 59600 tracks in their dataset with the following breakdown:

Genre	Number of Tracks
Classic Pop and Rock	23895
Folk	13192
Dance and Electronica	4935
Jazz and Blues	4334
Soul and Reggae	4016
Punk	3200
Metal	2103
Classical	1874
Pop	1617
Hip-Hop	434

As you can see, it's very unbalanced. They acknowledge this fact and also point out some other possible flaws. They say that the musicbrainz tags that were used to compile the dataset could be wrong or incomplete for some artists. This dataset isn't perfect but it is reflective of a lot of real world datasets. As for song features, they used the ones from the Echo Nest data that the Million Song Dataset is based on. Each song has information about loudness, tempo, time_signature, key, mode, duration, average and variance of timbre vectors. They also include the artists name, song titles, and a unique track id.

5.12 Lyric Dataset

The lyric data set is also a subset of the Million Song Dataset (MSD). It is composed of tracks from MSD. Each track's lyrics are represented in bag-of-words format; however, only the top 5000 words based on relative importance across the entire dataset are used. For our research, we used the musiXmatch dataset contained within the MSD. Due to the imbalanced nature of this dataset, we opted to leave out certain genres which had too few instances to datamine and yield useful results. Of the original list of genres, we ended up keeping the genres shown in the table below. Our dataset contained 34953 tracks in total.

Genre	Number of Tracks
Country	5064
Electronic	3624
Folk	1344
International	2155
Latin	5537
Pop Rock	5752
Rap	5291
RnB	4689
Vocal	1497

5.2 Evaluation Metrics

Due to how unbalanced the data is we decided against using accuracy as our evaluation metric. Accuracy assigns equal severity to all errors in classification. Instead we decided to use F1-Score.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

The F1-Score is the weighted average of Precision and Recall. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Recall is the ratio of correctly predicted positive observations to the all observations in actual class. Therefore, F1-Score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall. We used Scikit-Learn's F1-Score implementation. We set the 'average' parameter to 'micro' because our classes aren't binary.

5.3 Audio Feature Results

For the audio data we experimented with a few different algorithms. For each algorithm we used F1-Score for evaluation and K-Fold cross validation with 3 folds. The first algorithm we tried was K-Nearest Neighbors with 10 Neighbors. The test data got a 43.22% F1-Score and the cross validation scored a 43.52% F1-Score. Setting n=10 gave us the best results. Using more than 10 didn't really improve the result and less than gave a worse score.

Next we tried logistic regression. Initially, this gave use a small improvement over K-Nearest Neighbors. Using the Scikit-Learn logistic regression implementation gave us the ability to easily tune the parameters and perform many experiments. We set the 'penalty' parameter to 'l1', 'solver' to 'liblinear', and 'multi_class' to 'auto'. After tuning the parameters we were able to get a 56.16% F1-Score on the test data and 56.22% cross validation F1-Score.

The next model we thought would perform well was decision trees. We used a Scikit-Learn's DecisionTreeClassifier module and set the maximum depth of the tree to 11. The test data got an F1-Score of 49.71% and a cross validation F1-Score of 49.85%. This model seems to have suffered the most because of the unbalanced data from the first two. The tree was more biased to the majority classes.

Staying on the decision tree track, the final model we used was random forest classification. This was the one we spent the most time with and also the best performing model on the audio data. A random forest fits a number of decision tree classifiers on many random samples of the data and averages the results to improve the accuracy. A huge advantage of using random forest is they correct for over-fitting that you can get with singular decision trees. We used a Sckit's random forest implementation the 'n_estimators' parameter set to 250 (250 decision trees). The random forest approach scored a 58.00% F1-Score for the test data and 58.67% for cross validation.

Model	F1-Score	CV Score
Random Forest	58.00%	58.67%
Logistic Regression	56.16%	56.22%
Decision Tree	49.71%	49.85%
K-Nearest Neighbors	43.22%	43.52%

5.32 Lyric Results

Like with the audio data, we also experimented with a handful of classifiers. Of the 34953 music tracks in the dataset, 300 tracks from each genre were selected for use in the test set, the remaining tracks were used to train the classifier being tested. For each implementation we used F1-score as our evaluation metric. Like before, we also performed 3-fold cross validation. From this, we determined that the parameters which gave us the best results for each classifier were as follows:

Linear SVM Classifier Parameters: C = 10.0 and max_iterations set to 5000.

This was by far the most successful approach which we tried on this dataset. Unfortunately, due to the sheer length of time required to run this classifier with larger max iteration values we decided against trying even larger values past 5000. This would be something we would explore if we had the opportunity to continue our experimentation. Refer to figure 4 below, to see the results generated from our Linear SVM implementation.

Linear SVC	**precision**	**recall**	**f1-score**	**support**
Country	0.45	0.88	0.6	300
Electronic	0.7	0.58	0.64	300
Folk	0.97	0.48	0.64	300
International	0.94	0.74	0.83	300
Latin	0.82	0.97	0.89	300
Pop_Rock	0.62	0.51	0.56	300
Rap	0.86	0.86	0.86	300
RnB	0.64	0.64	0.64	300
Vocal	0.85	0.82	0.83	300
AVERAGE/TOTAL	0.761111111	0.72	0.721111111	2700

Figure 4.

Complement Naive Bayes Classifier Parameters: alpha = 1.0

Surprisingly, this ended up being the worst performing classifier that we tried. Initially, when we first tried this, the results were very good and almost matched those from the SVM classifier. However, when we made the decision to

remove some of the classes of data due to the fact that there simply were not enough instances to generate a meaningful result, the Naive Bayes Classifier did not perform very well despite us running gridsearch cross validation.
Refer to figure 5 below, to see the results generated from our Naive Bayes implementation.

Naive Bayes	**precision**	**recall**	**f1-score**	**support**
Country	0.28	0.54	0.37	300
Electronic	0.75	0.01	0.02	300
Folk	0	0	0	300
International	0.63	0.24	0.34	300
Latin	0.59	0.96	0.73	300
Pop_Rock	0.39	0.32	0.35	300
Rap	0.46	0.78	0.58	300
RnB	0.28	0.71	0.4	300
Vocal	0	0	0	300
AVERAGE/TOTAL	0.375555556	0.395555556	0.31	2700

Figure 5.

Decision Tree Classifier Parameters: max_depth = 10, max_features = 1000
One interesting observation which we made after trying this approach was that this actually performed better than the Naive Bayes implementation. In our previous research and experience, typically for tasks involving text classification, a naive bayes classifier outperforms a decision tree. Something which we would have liked to do was to create an even more sophisticated pipeline for feature extraction to improve the results of this classifier.
Refer to figure 6 below, to see the results generated from our Decision Tree implementation.

Decision Tree	**precision**	**recall**	**f1-score**	**support**
Country	0.24	0.67	0.35	300
Electronic	0.49	0.09	0.16	300
Folk	1	0.01	0.01	300
International	0.53	0.5	0.52	300
Latin	0.78	0.88	0.83	300
Pop_Rock	0.23	0.5	0.32	300
Rap	0.76	0.68	0.72	300
RnB	0.51	0.41	0.45	300
Vocal	0.8	0.03	0.05	300
AVERAGE//TOTAL	0.593333333	0.418888889	0.378888889	2700

Figure 6.

K-Nearest Neighbor Classifier Parameters: n_neighbors = 13

In an attempt to incorporate what we learned in class, we also decided to include this classifier in our experimentation. Another surprising discovery we made was that this classifier outperformed both the decision tree and naive bayes implementations. Once again, if we were to continue working on this project, we would like to try and improve our feature selection pipeline in an attempt to improve the results of this classifier.
Refer to figure 6 below, to see the results generated from our K-Nearest Neighbor implementation.

K Nearest Neighbor	precision	recall	f1-score	support
Country	0.36	0.52	0.42	300
Electronic	0.3	0.35	0.32	300
Folk	0.45	0.08	0.13	300
International	0.56	0.39	0.46	300
Latin	0.72	0.92	0.81	300
Pop_Rock	0.24	0.61	0.35	300
Rap	0.64	0.67	0.65	300
RnB	0.59	0.32	0.42	300
Vocal	0.54	0.04	0.08	300
AVERAGE//TOTAL	0.488888889	0.433333333	0.404444444	2700

Figure 7.

6 Conclusion

In this project we used extracted audio feature and lyric data to classify music genres. Through our testing we found that both Random Forest and Support Vector Machine classifiers yielded the best performance on the audio and lyric datasets respectively. Throughout the course of this project we had to overcome various challenges related to the datasets themselves. For example, as we discussed above, the audio dataset suffered a class imbalance problem which tended to skew our classifiers toward one class more than the rest. Our solution to this was to adjust the weights of each class to compensate. Another challenge we faced, was that the lyric dataset did not have any genre labels, to solve this we generated a new dataset by combining multiple MSD datasets which each had information we needed. Luckily, all of the datasets use one single primary key which is consistent across the entire MSD dataset cluster which allowed us to create a dataset which contained the information we needed. As a result of this experience, we now have even more experience dealing with imbalanced data. In the end, we were very pleased with the results since we were able to achieve much success with both Random Forests and SVMs which mirrored the results of other attempts that we researched.

6.1 Directions for Future Work

If we were to continue work on this project, some things which we would pursue building our own more balanced and complete datasets and trying to do genre classification on the raw audio signal data as opposed to lyrics or audio metadata. In addition to this, we would also like to try combining both lyric data and audio data into one dataset and attempt to perform genre classification on that.

7 Reference

1. MSD Genre Dataset
   https://labrosa.ee.columbia.edu/millionsong/blog/11-2-28-deriving-genre-dataset

2. MSD Lyric Dataset
   https://labrosa.ee.columbia.edu/millionsong/musixmatch

3. Xu, Changsheng & Maddage, Namunu & Shao, Xi & Cao, Fang & Tian, Qi. (2003). Musical genre classification using support vector machines. 5. V - 429. 10.1109/ICASSP.2003.1199998.
   https://www.researchgate.net/publication/4015150_Musical_genre_classification_using_support_vector_machines

4. A. Goulart, R. Rodrigo, and C. Maciel, "Exploring different approaches for music genre classification," *NeuroImage*, 18-Apr-2012. [Online]. Available:
   https://www.sciencedirect.com/science/article/pii/S1110866512000151#