Classification of Artist Genre through Supervised Learning

Richard Ridley and Mitchell Dumovic

Abstract – **The goal of this paper is to classify the genre of an artist given a set of quantitative measures for each of their associated songs. We utilized supervised learning in this task, and relied upon a dataset composed of quantitative song data that Spotify provides for the majority of the songs that they stream. We collected data for 14 different features for over 35 million songs, which spanned over 100 thousand artists and over 12 hundred genres. Before applying classification methods to this data, we collapsed our set of genres by a factor of ten, and collapsed our song data by applied different statistical measures to reduce to a set of features for every artist. Then, we applied Stochastic Gradient Descent, Gaussian SVM, and Nearest-Neighbors as classifiers, and results were somewhat successful.**

INTRODUCTION

Genre classification is immensely important to the field of music. Accurate genre classification can aid in the effectiveness of music recommendation engines, help unearth similarities between different music genres, and reduce the need for the hand labeling of genres in streaming services. This project focuses on attempting to accurately predict the genres that an artist belongs to given information about the songs that they have produced. The input to our algorithm is a dataset that we pulled from Spotify containing information about artists (with their associated list of classified genres) and tracks (with their associated song features). We then use various classification algorithms (KNN, SGD, SVM) to output predicted genre classifications for artists.

RELATED WORK

Most related works we found dealt with song classification instead of artist classification, but we still were able to get valuable ideas and information from various sources.

For feature extraction, [2] and [4] used "Spectral Centroid, Spectral Roll-Off, Spectral Flux, Time Domain Zero Crossings, Pitch Distribution and 13 different Mel-Frequency Cepstral Coefficients of individual tracks" whereas [5] and [1] deal with extracted features related to timbre, melody, rhythm, and pitch. In general, most papers we found pulled features from the raw song waveform, so we knew that we would have to extract features derived from these waveforms as well. However, we found that some of these approaches like [4], [5], and [6] suffered from relatively small datasets or limited numbers of genre labelings.

In terms of models, the model that performed best in [4] was k-nearest neighbors applied to the multi-label data. [5] suggests using support vector machines as well as attempting one versus one and one versus all classification algorithms for the multi-label data. [6] suggests the use of an ensemble technique for multi-label classification, another classification algorithm which involves training multiple classifiers for the data and combining the results of the multiple classifiers into one single classification. In general, it became quickly apparent from these related works that we would need to use one of the multi-label methods for classifying our data, and would really benefit from attempting algorithms involving the training of multiple classifiers, as these seemed to perform best over the similar papers that we looked at.

DATASET

To collect our data, we relied upon a publically available Spotify API that allowed us to capture information about specific songs, artists, and their albums. To collect data about the songs of a wide range of different artists, we first needed to assemble a set of artists for which we could collected song data for. To do so, we first hand-picked a "root" set of 50 different artists that we believed to be representatives of a wide variety of

different genre clusters. After doing so, we used a functionality that would allow us to find all of the artists related to a specific artist. Through the use of recursive formulation, we assembled a list of over 100,000 artists that would form the basis for our training data. Each of these artists had labeled genres, which indicated to which genres they were a part of. Then, for each of these artists, we collected a set of all of their associated songs. For each of these songs, we were then able to collect a variety of different quantitative measures over different qualitative aspects of our songs. The features that we collected are below.

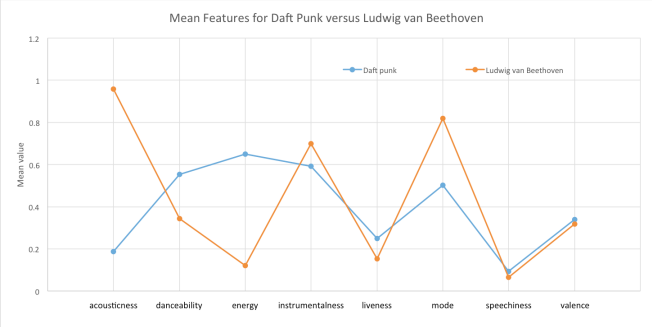
AVAILABLE SONG FEATURES

Feature	***Description***
acousticness	A confidence measure of how acoustic the track is.
danceability	A metric assessing the degree to which the track is danceable.
energy	Represents a perceptual measure of the songs intensity and activity.
instrumentallness	A confidence measure dictating whether the track contains vocals.
key	An integer indicating the key the track is in.
liveness	A measure that details the likelihood that the track was performed live.
loudness	The overall loudness of the track as measure in decibels.
speechiness	The confidence in the appearance of spoken words in a track.
tempo	The overall estimated tempo of a track in beats per minute.

FEATURES

Because songs do not have a genre labeling, we needed to build a feature set for every artist that accurately represented the set of songs associated with them. And since artists could have a highly variable number of songs, we needed to develop a strategy that would allow us to have a static number of features defined over a variable number of input songs. To do so, we calculated different statistical measures over each of our individual song features for each of our artists. We decided to use mean, median, variance, and skew.

This succeeded in reducing the dimensionality of our data set, and also reduced our training set to about 100,000 examples, consisting of agglomerated song data for every artist that we had collected. However, this did not come without a cost, as the use of these summary statistics reduced the precision of the data that we had collected by a significant degree. Nonetheless, artists of different genres generally differed significantly in their values for these statistics, as the below example illustrates.



GENRE COLLAPSING

In our original dataset, Spotify classified songs with 1241 unique genres. As some of our algorithms involve training classifiers for each genre labeling, it was necessary to try to reduce the number of genre labelings as much as possible. To do this, we relied heavily on association rule algorithms in order to find labelings that appeared frequently together. These algorithms worked roughly as follows:

First, define *support* for some set of genres G - $supp(G)$ - as the number of times that an artist is classified with all the genres in the set G. Our algorithm started by removing genre labelings for single genre set below a certain support threshold over the entire dataset. Next, define *confidence* and *lift* for some pair of genres G_1 and G_2 as follows:

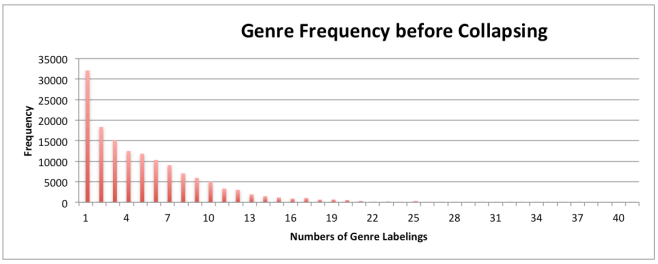
$$conf(G_1 \rightarrow G_2) = \frac{supp(G_1 \cup G_2)}{supp(G_1)}$$

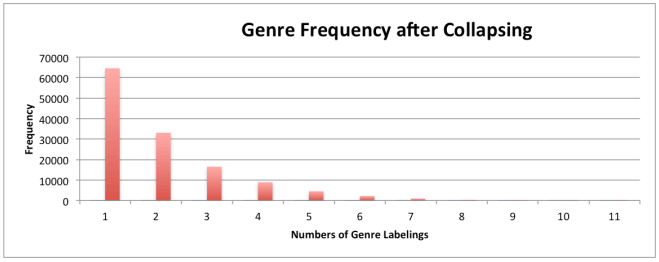
$$lift(G_1 \rightarrow G_2) = \frac{supp(G_1 \cup G_2)}{supp(G_1) * supp(G_2)}$$

A confidence value of 1 indicates that every time that G_1 appears it appears with G_2,

meaning that it is a good candidate for genre collapsing. For example, in our analysis we found that the genre "college a capella" appeared together with the genre "a capella" the exact same number of times the genre "college a capella" appeared by itself. This indicates a confidence value of 1, and resulted in us collapsing "college a capella" together with "a capella."

In general, our algorithm worked by iteratively collapsing the two genres with the highest confidence and lift values in our dataset, terminating when the confidence and lift values fell below a certain threshold. Finally, we scanned over the entire dataset once more and removed any more collapsed genres that had a support value lower than a second, higher threshold. This resulted in our final collapsed list of 95 unique genres. This also had the added effect of reducing the number of labelings an artist was classified as on average, further simplifying our dataset. Before this genre collapsing, each artist was classified with an average of approximately 5.27 different genre labelings. After the collapse, this number was reduced to an average of approximately 1.88 different labelings.





CLASSIFICATION MODELS

To classify the genres for different artists, we used the supervised learning approaches of Stochastic Gradient Descent, K-nearest neighbors, and Support Vector Machines. In configuring Stochastic Gradient Descent and Support Vector machines in this multi-label classification problem, we decided to try using both "one versus one" and "one versus all" approaches.

A "one versus all" approach in the context of multi-label classification problem involves the training of a binary classifier for each of the labels. This classifier discriminates between its associated label and all of the other labels in the training set. In our case, when using this approach, we trained 95 different classifiers which would tell us whether an artist either belonged to or did not belong to a specific genre, and labeled that artist with that genre if the classifier outputted that the artist does belong to that genre.

We also employed a "one versus one" classification scheme when using SVM. "One versus one" in the context of multi-label classification involves the training of a classifier for every pair of labels, and learns to distinguish which of the two labels an example is more likely to belong to. After training these classifiers, an example is assigned a label according to the class that got the highest number of positive predictions. Thus, in our case, we trained a classifier for every pair of musical genres, and when evaluating what genre that an artist belonged to, we returned the genre that was predicted most often in the 95 pairs in which it appeared. So as to ensure that artists could attain multiple genre labelings, we assigned additional genre labelings if the number of times that these individual genres were predicted was close to the number of times the mostly commonly predicted genre was predicted.

Both "one versus one" and "one versus all" can be considered valid, but it is generally the case that "one versus one" multi-label classification takes significantly more processing time and requires more data than does "one versus all" approaches.

Multi-label K-Nearest Neighbors Model

When attempting to classify a testing example x, let A be a sorted list containing the Euclidean distances of the training examples to the test point. Let $A_{i,\,y}$ denote the label of the training example

that is the i-th closest in terms of Euclidean distance to the test point. The single label k-NN algorithm uses the following to output select the output label:

$$testLabel = \ arg_k \max \left(\sum_{i=1}^{k} 1[A_{\{i,y\}} = k] \right)$$

In the multi-label algorithm, the neighbors are first found, then a maximum a posteriori (MAP) principle is utilized to determine the test label[7]. Since we operated in a space with so many possible labelings, we set k to be equal to 10.

SVM Model

The SVM model is a model that seeks to minimize an objective function by creating a hyperplane that is used to separate two classes of data inputs and maximize the separation between both sets of points and the hyperplane. We utilized a soft margin with a hinge loss function, such that if a data point is classified on the wrong side of the hyperplane, its Euclidean distance from the hyperplane is added to the objective function.

$$\min \left(\left[\sum_{i=1}^{n} \max\left(0, 1 - y_i(w \cdot x_i - b)\right) \right] + \lambda \|w\|^2 \right.$$

The parameter λ determines the tradeoff between increasing the margin-size and ensuring that data points lie on the correct side of the margin.

A point *x* is classified as having label -1 if

$$w \cdot x - b = 1$$

and having label 1 if

$$w \cdot x - b = -1$$

where b denotes the size of the margin chosen to separate the two classes of data points from the separating hyperplane.

Stochastic Gradient Descent

Stochastic gradient descent is a machine learning approach that can be used to quickly learn a classifier that can distinguish between two classes of data points. It works by updating its classification model, which is represented by a weight vector, over every data point that it observes, and suspending its iteration after a convergence to a local minimum, or after some number of passes through its provided training data. The update rule is below.

$$w \leftarrow w - \eta\left(\alpha \nabla_{\mathrm{w}} \mathrm{R}(\mathrm{w}) + \nabla_{\mathrm{w}} L(wx^i, y^i)\right)$$

The function *R* is a function chosen is called the regularization term, and acts to minimize the size of the weight vector *w*. We used a reward function equal to the L2 norm of the weight vector.

The function *L* is called the loss function, and provides a measure of the difference between an examples classification and its actual class. We used a logistic regression lost function.

$$L_{\log}(wx, y) = \log\left(1 + \exp(-y \cdot w^T x)\right)$$

The learning rate, η, scales how quickly the classifier "learns", or is updated. We used an annealing learned rate, meaning that that the learning rate decreased with the number of iterations, thus ensuring that our classifier converged quickly.

RESULTS

In the end we trained using a training set consisting of 21,838 artists and a test set of 2,426 artists. Our results from our different approaches in the confusion matrix form are below. Since, artists can have multiple genres, the term y_{pred} denotes whether a specific genre was predicted for a specific artist. Similarly y_{obs} denotes the presence of specific genre within an artist's classifications.

	Test Error	
K-nearest Neighbors	$y_{pred} = 1$	$y_{pred} = 0$
$y_{obs} = 1$	.023	.977
$y_{obs} = 0$	.032	.968

	Train Error		***Test Error***	
SGD	$y_{pred} = 1$	$y_{pred} = 0$	$y_{pred} = 1$	$y_{pred} = 0$
$y_{obs} = 1$	.097	.903	.067	.933
$y_{obs} = 0$	.020	.980	.031	.969

	Train Error		***Test Error***	
SVM	$y_{pred} = 1$	$y_{pred} = 0$	$y_{pred} = 1$	$y_{pred} = 0$
$y_{obs} = 1$	0.162	0.838	.115	.885
$y_{obs} = 0$	0.026	0.974	.028	.972

As expected our SVM approach performed the best. However, in general, we were disappointed with our results. Our hit-rate was much smaller than we originally expected. However, this was in large part due to the high dimensionality of our data: with 95 possible genre labelings, a hit-rate of 11.5% is far better than randomly choosing genres. Additionally, we found that we had a better hit rate on some genres than others: our algorithm in general had much more hits on niche genres like classical, metal, and deep electronic music than more generic genres like rock. Below are confusion matrices for the collapsed genres roughly corresponding to the classical and heavy metal music genres:

	Test Error	
Classical	$y_{pred} = 1$	$y_{pred} = 0$
$y_{obs} = 1$	.512	.488
$y_{obs} = 0$	.017	.983

	Test Error	
Heavy Metal	$y_{pred} = 1$	$y_{pred} = 0$
$y_{obs} = 1$	.452	.548
$y_{obs} = 0$	.021	.979

As it was clear that our feature set was effective in differentiating clearly unique genres, we decided to train a classifier on a reduced training set containing only artists having the genres of classical, heavy metal, deep electronic, or country. We constructed a similar testing set, and ended up with a training set of 1686 examples, and a testing set of 484 examples. We utilized a one versus one SVM with the same hyper parameters as before. Its confusion matrix is below.

	Test Error	
Reduced Data Set	$y_{pred} = 1$	$y_{pred} = 0$
$y_{obs} = 1$	.678	.322
$y_{obs} = 0$	.125	.875

As expected, it performed very well relative to the classifier trained and tested on our whole data set. Such a result was heartening, and seemed to display that the features we collected could be relevant in helping to classify the genres of artists.

CONCLUSION

Our best performing algorithm was SVM using a one-versus-one scheme, followed by SGD using a one-versus-all scheme, followed by multi-label k-nearest neighbors. In general, however, we were a bit disappointed with the hit rate of our results. The two main reasons we believe that our results over our entire training set were poorer than anticipated are due to the high dimensionality of our data and our base features used. The need to use an algorithm to collapse genres together distorted our original data, and the 95 genres we were left with were still far too many to train an effective classifier. Additionally, while the Spotify song features may be useful for recommendation purposes, it is likely that by aggregating them using different statistical measures we lost a lot of the information that makes genres unique.

NEXT STEPS

If we were to work improve our results, we would definitely start by improving the quality of our training set. This would involve reinventing our feature selection and extraction algorithms as well as our genre collapsing algorithms so as to both reduce the dimensionality of our problem and provide the best possible features for prediction. Additionally, we may also look at alternative methods at classification, and perhaps first build classifiers for the genre of individual songs, which could be used to classify artists. Such a scheme would allow us to operate directly on all of the data that we collected, rather than operating on summary statistics and measures of our data.

REFERENCES

[1] N. Scaringella, G. Zoia and D. Mlynek, "Automatic genre classification of music content: a survey," in *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 133-141, March 2006.

[2] McKinney, Martin F., and Jeroen Breebaart. "Features for Audio and Music Classification." *Features for Audio and Music Classification*. Johns Hopkins University, n.d. Web. 15 Dec. 2016.

[3] Prasanna, K., and M. Seetha. "ASSOCIATION RULE MINING ALGORITHMS FOR HIGH DIMENSIONAL DATA – A REVIEW." *International Journal of Advances in Engineering & Technology* (2012): n. pag. Web.

[4] Silva, Vitor Da, and Ana T. Winck. "Multi-Label Classification of Music into Genres." *Applied Data Mining* (2013): 181-203. Web.

[5] Wang, Shu. "Musical Genre Categorization Using Support Vector Machines". N.p., 2016. Web. 12 Dec. 2016.

[6] Sanden, Chris, and John Z. Zhang. "Enhancing Multi-label Music Genre Classification through Ensemble Techniques." *Proceedings of the 34th International*

[7] Zhang, M.L.; Zhou, Z.H. (2007). "ML-KNN: A lazy learning approach to multi-label learning". *Pattern Recognition*. **40** (7): 2038–2048.