MUSIC GENRE CLASSIFICATION: NEAR-REALTIME VS SEQUENTIAL APPROACH

A PREPRINT

Scott Zhang
zhangsc@stanford.edu

Huaping Gu
hpgu@stanford.edu

Rongbin Li
rongbli@stanford.edu

June 12, 2019

ABSTRACT

***K*eywords** Music Genre Classification · Machine Learning · RNN · MFCC · SKLearn · keras

1 Introduction

In this project, we use a selection of algorithms from the class CS229 to classify music into different Genres. To get reliable performance bench-marking, we use the industry famous GTZAN Genre Collection dataset. We start with calculating the classifications with several traditional classic algorithms, then compare with some modern deep learning algorithms with heuristics layer design and data augmentation. We propose an unique near real-time music genre classification solution, by listening a music 0.5 seconds we can identify its genre with 64% accuracy. We conclude that RNN is powerful in classifying a longer track, while even short time MFCC features can already get decent classification accuracy. Using the mean and covariance of MFCC as an augmented feature set is inspired by Million Song Dataset (corresponding to the field section_timbre), and its subset in UCI repository.

2 Related Works

Music genre is a very hot topic in machine learning, not because of its commercial business value, but of the unique attributes of the music which natively contains lot of digital music metadata. Lot of researches were already been applied on this field. The dataset we used are for the well known paper in genre classification " Musical genre classification of audio signals " by G. Tzanetakis and P. Cook in IEEE Transactions on Audio and Speech Processing 2002.

Some work in [1] and [2] also did lot of research on music genre classification. Especially the [2] work claims can get 91% accuracy of on the GTZAN[3] datasets. Lee [4] and his team did some work on convolutional deep belief networks for unsupervised audio classification.

3 Datasets

We are using the GTZAN Genre Collection dataset. GTZAN has total 10 different genres, each containing 100 audio clips that are 30 seconds long. genres, each containing 100 audio clips that are 30 seconds long. We use librosa Python library to extract features from the wave files. Librosa will return a 2D array. e.g. if we use Mel-frequency Cepstral Coefficients (MFCC) we will get one (12×1293) array for a 30 seconds 220 Hz music with hop-length=512. x coordinate is time and y coordinate is 1 of the 12 coefficients. In our project, we will use two librosa methods to extract the raw data from the wave file, chromagram and MFCC, of the same shape. Bellow are plotted output for two genre 2D arrays.

4 Methodology

We use librosa package to transform the wave file into mel-spectrograms (MFCC) and chromagram, both of which are 2D array in terms of time and feature value. e.g. for MFCC, the x is time while the y is the mel-frequency. This is a standard approach to processing music and speech, because mel-scale corresponds well with human sound perception. Our approach basically has two folds.

In the first fold, we use the basic feature extraction (Algorithm 1) which get a $n \times 14$ 2D array. Then we use the classic model applied to MFCC and chromagram datasets (Basic Benchmark).

In the second fold, we first do data normalization, then extend our feature extraction with the covariance (Algorithm 2), which augment the basic dataset into a $n \times 92$ 2D array. With these augmented datasets, we re-test all the classic models and some new modern models includes CNN and RNN.

In our approach, we use the mean and covariance of MFCC as an augmented feature set, which is inspired by Million Song Dataset (corresponding to the field section_timbre), and its subset in UCI repository.

Finally we plan to answer questions:

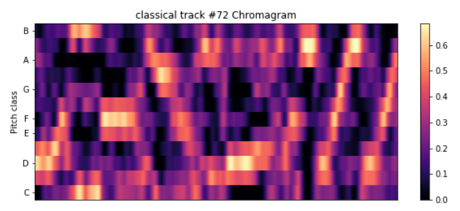
- How important is the time dimension in music genre classification? For example, what if we simply take the 1d array of power with respect to mel-frequency, taking average in time, how much worse off will the prediction be? This really should be the natural way of dimension reduction. This will also serve as a computationally cheap sub-baseline.

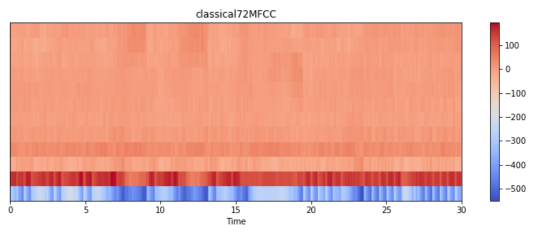
- Whether we can simply use a very short period music to predict the genre? And what is the accuracy? And how could improve the performance of prediction using time-series techniques?

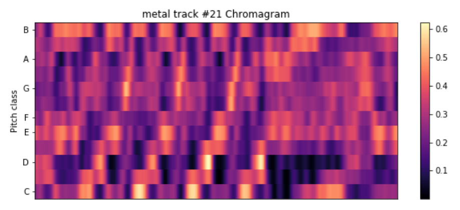
To avoid over-fitting, we will employ various cross-validation techniques, such as k-fold cross-validation and shuffling.

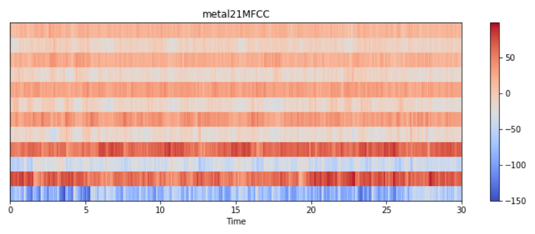
5 Feature Extraction

We use librosa API to extract wave file, usually librosa will transform it into an 2D array. e.g. if we use mel-spectrograms (MFCC) we will get one (12 x 1293) array. Since in image recognition, 2D CNN is common, because we want to detect features invariant under 2D translations by “slide a small window over the image”. When we are dealing with music, x coordinate is time and y coordinate is mel-frequency. In our project, we will use two librosa methods to extract the raw data from the wave file, MFCC and chromagram. Bellow are plotted output for two genre.









However, it is not clear that translation in the frequency direction makes sense: a music pattern in high pitch vs low pitch corresponds to very different information. In other words, a small time and frequency window is not symmetric across 2D. By contrast, from human heuristics, although music progression is important, we expect to identify a music genre from any short clip (1 to 2 seconds).

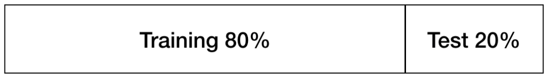
We investigated several methods about how to transform from 2D to 1D. We finaly use one cheaper version of 1D is to cut up each song into short clips, to create more training/validation/test examples. From heuristics, hearing a musinc longer should give more confidence to guess the music genre. MFCC relates more to timbre, so in some sense maybe time progression is less important? Chromagram is related to melody/harmonics, maybe more dependent on time. We extracted both and did some research and evaluate the differences as parts of our project.

6 Basic Benchmark

First we will start a simple solution on feature extraction, and some well know classical training models. We will use the outputs from these models as the benchmark for next step analysis.

6.1 Training and Test Datasets

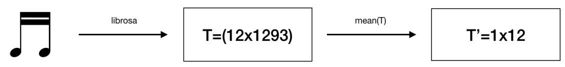
In the GTZAN collection, we have totally 10 genre and each has 100 songs. In this basic benchmark, we will split the dataset into two parts, 80% used for training, and the rest of the 20% are used for the final testing.



To make sure the training data are balanced, we make sure pick 80% (80 songs) from each genre for training.

6.2 Benchmark Features

In our basic benchmark, for one single sample (one song), first we extract data from wave file into a 2D array by librosa API. To make the basic benchmark simple and reliable, we calculate the $mean()$ on the x time coordination, transverse the array, we finally get a (1 x 12) 1D array for each sample.

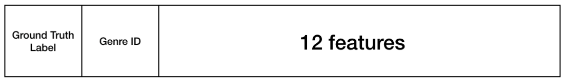


We run bellow algorithm with all the sample wave files,

```
Result: Extracted song features (n ×14)
initialization;
while $i$ < $n$ do
    librosa$_i[12 \times 1293] = load\_by\_librosa()$
    feature$_i[1 \times 12] = mean\_by\_time()$
    features$_i[1 \times 14] = fill\_GT\_genre\_id$
end
return features;
```

Algorithm 1: Benchmark Features

At the end end of the algorithm, we got a (12 x n) 2D array. Then we combine these features with the ground truth (extract from the folder structure), get an (14 x n) 2D array,



We use sklearn framework to apply to several classic machine learning algorithms.

6.3 Benchmark Results

We tried two librosa feature extraction methods, one is chromagram and the other is MFCC.

- Chromagram: chromagram closely relates to the twelve different pitch classes
- MFCC: Mel Frequency Cepstral Coefficient (MFCC), used for for identifying the linguistic content and discarding all the other stuff which carries information like background noise, emotion etc. Usually this related to the musical instrument.

After we got the features, we applied them to the classical algorithms includes bunch of logistic regression algorithms and SVM models.

We used 4 Logistic regression algorithms with multinomial option on, all of them we use the l_2 error.

- "newton-cg": newton method
- "lbfgs": an optimization algorithm in the family of quasi-Newton methods that approximates the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm using a limited amount of computer memory
- "sag": Stochastic Average Gradient
- "saga" : A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives

And used two SVM algorithms

- SVC: Support Vector Classification
- LinearSVC: Linear Support Vector Machine for classification

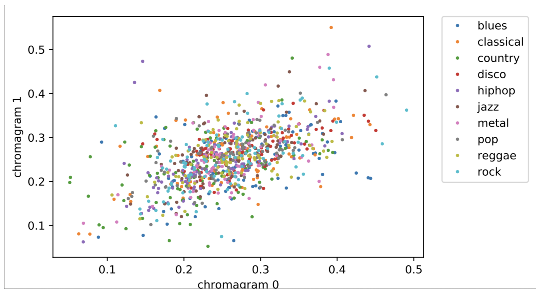
We got following results:

Features/Algorithms	MFCC	Chromagram
Newton-cg	51.5%	19%
lbfgs	51%	19%
sag	49%	19%
saga	50%	19%
SVC	42.5%	30%
LinearSVC	35.5%	24.5%

6.4 Benchmark Analysis

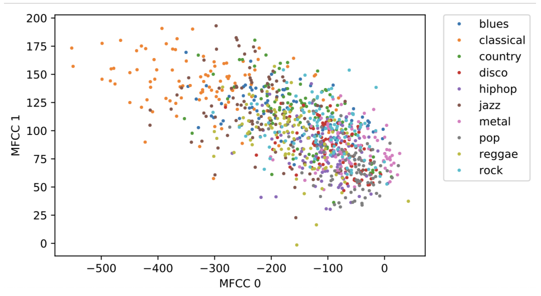
From the benchmark results, we found that MFCC performs much better than the Chromagram. Based on the definitions of MFCC and Chromagram, it is not surprise to see this results. In the benchmark data extraction step, we did $mean()$ on the sample's dataset, we did lose some of the information.

From Chromagram, since it is related to the twelve different pitch classes, which are very common and used across all different genre. And after we do the $mean()$, we cannot see the differences between genre. Bellow plot on chromagram dataset shows there is no clear classification boundary for 10 genrs.



While when we plot the MFCC for the first and second Coefficient, we can see some genre and very well been separable. This does tell us we need to do more enhancement in the feature extraction step.

Like this bellow plot, Genre classical, metal and reggae are very well separable.



7 Data Normalization and Augmentation

7.1 Data Normalization

We do pre-processing by standardizing the ex- tracted features, placing the input features in the same scaled space. We sometimes add L2 regularization while training, to avoid overfit- ting. On average, with the standardization and regularization, we can see some % enhancements on the final test accuracy on all algorithms above, especially the SVM algorithms on MFCC.

Features/Algorithms	MFCC	Chromagram
Newton-cg	53.4%	25.8%
lbfgs	53.2%	26.0%
sag	51.0%	25.8%
saga	51.4%	25.8%
SVC	61.5%	28%
LinearSVC	50.0%	25.0%

7.2 K-Fold Experiments

Because of the small datasets size($100x10 = 1000$) we noticed our algorithms over-fitting. By average we always get about 5-20% gap between the training accuracy and test accuracy. One way we tried is to use the K-Fold with K=5. We split the training datasets into 5 slots, and rotate $K - 1$ slots as training set and the left slot as validation set. Since we already confirmed MFCC has much better results than Chromgram, we applied K-Fold on MFCC datasets only and test on logistic algorithms.

Algorithms with MFCC	Training Acc.	Test Acc.
Newton-cg	58%	56.25%
lbfgs	56.75%	55.38%
sag	53.38%	53.38%
saga	53.63%	53.63%

But finally we decided to use the "sample augmentation" by which we segmented the current 30 seconds music into multiple parts by various split internals. Since we got enough training sets , we did not move on this K-Fold solution.

7.3 Data Augmentation

In our basic bench-marking step, we use left algorithm $n \times 14$ 2D array. But in the enhanced feature extraction, we append extra 78 co-variance features, totally we get a $(n \times 92)$ 2D array.We use these extended features in all the other tests, including CNN and RNN.

```
Result: Extracted song features (n×92)
initialization;
while $i < n$ do
    librosa$_i[12 \times 1293] = load\_by\_librosa()$
    feature$_i[1 \times 12] = mean\_by\_time()$
    features$_i[1 \times 14] = fill\_GT\_genre\_id$
    features$_i[1 \times 92] = ext\_cov\_78(feature_i)$
end
return features;
```

Algorithm 2: Extended Features

8 Classical Model on Segmented Dataset

We segmented the 30 seconds music into multiple small parts. For example, if we divide each song into 30 segments (each has 1-second long), we would totally get $30,000$ samples, and for each segmented sample, the size is $1 \times (90 + 1)$ (90 features plus 1 label). We run classical model (multiclass LogisticRegression, SVC, LinearSVC) on the segmented dataset, get the results on the right. From the results, we can see that, the best test acc we can get for segmentated data is about 70%, but there is still obviously overfitting. To get a better result, we need to consider time series and apply a sequential approach (see sections below).

Models	Seg	Training Acc.	Test Acc.
LR	3	78.16%	68.16%
LR	10	70.97%	64.60%
LR	30	63.39%	59.91%
SVC	3	92.20%	71.66%
SVC	10	90.77%	69.25%
SVC	30	86.61%	66.28%
LinearSVC	3	82.95%	64.50%
LinearSVC	10	71.46%	63.4%
LinearSVC	30	63.73%	59.03%

9 Modern Models on Segmented Dataset

9.1 Fully Connected Neural Network

After we get the augmented datasets, we feed them to our deep learning framework. For NN we use the de faco industry standard framework $tensorflow.keras$. For all the datasets used bellow are all MFCC with 90 features ($(12 + 78)$).

We start with bellow NN network settings

Layer	Input Dim	activation	regularizer
1	90	relu	$l_2(0.01)$
2	1	softmax	-

We run 200 epochs with $batch\_size = 800$ and $optimizer =' rmsprop'$, with various segmentation intervals:

segments	Train Accuracy	Test Accuracy
1	70.88%	63.00%
10	74.14%	67.00%
20	69.95%	65.25%
30	67.34%	62.70%

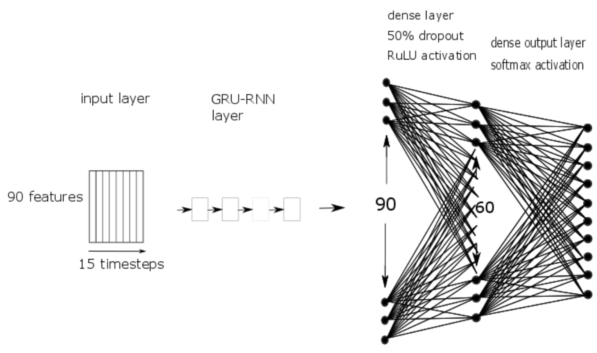
9.2 Recurrent Neural Network

RNN layer takes each time series as an input, and feeding it into a reappearing cell sequentially, each step updating an internal memory and the output. After reading all steps of the time series, the layer returns the final output of some dimension, say 90. GRU (grated recurrent unit) is one type of RNN cell, where each cell has a update gate and a reset gate. We use GRU in our RNN layer because of its speed and similar performance compared to LSTM (long-short term memory) units.

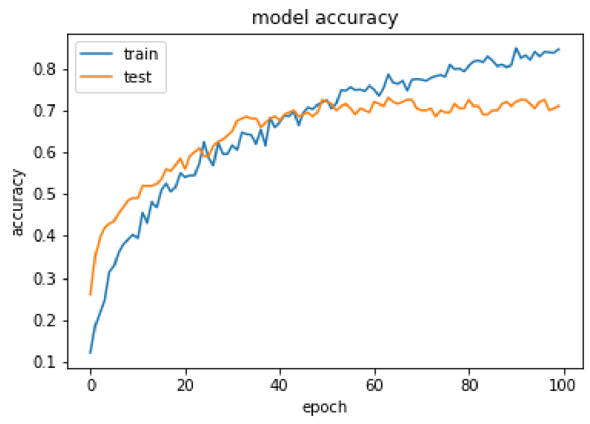
We implemented RNN using

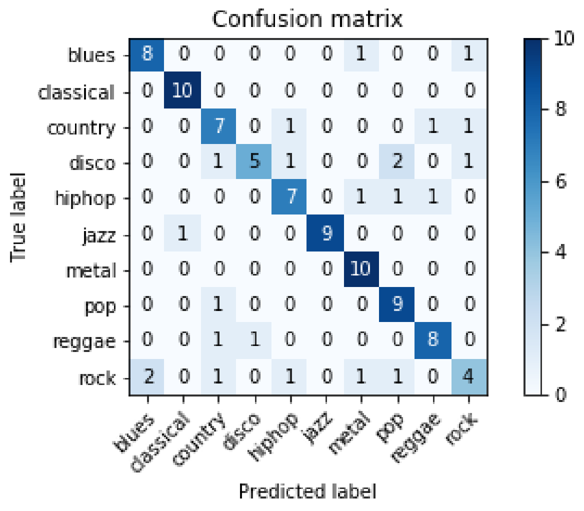
- An input layer of shape (15, 90), i.e. segmenting each audio into 15 timesteps.
- A layer of GRU with 90 output units, with 50% dropout rate and 10% recurrent dropout rate,
- A fully connected layer of 60 units, with ReLU activation, 50% dropout rate, and L2 regularization,
- An output layer of 10 units, with softmax activation.

9.2.1 RNN Architecture



9.2.2 RNN Test Results





We can see, load the MFCC datasets with $mean()$ and $covarance$, a LSTM-RNN can boost the test accuracy to near 80%.

Segs	GRU	Dense	Dropout	Train Acc	Test Acc
15	90	60	50	84.57%	77%
30	60	40	20	95.71%	72%
30	90	60	50	83.71%	75%
60	90	60	50	84.14%	70%

10 Conclusions

We find that MFCC features are relatively constant over time, so that even a short 0.5 second clip of music can be classified into 10 genres with more than 65% accuracy consistently. Taking MFCC average and covariance over longer period of time will improve the accuracy up to 70%, even for classical algorithms such as SVC. Treating MFCC average and covariance as a time series with time-step 0.5 second, a LSTM-RNN can boost the test accuracy to near 80%, proving that the time progression of MFCC indeed has predictive power in classifying genre.

11 Future Works

In our project, although we do the dataset augmentation by split the 30 seconds music into mutiple clips with various split segmentation intervals. But we still see some over-fitting and inconsistency in the accuracy results. More test data will absolutely help us move further and understand more deeper of the algorithms we learnt in this course.

We believe that more systematic cross-validation and hyper-parameter tuning could potentially improve the overall performance of our models.

Part of the reason why we are prone to overfitting and cannot further raise prediction accuracy is because of the small sample size of the GTZAN dataset. An interesting next step project will be either to scale our approach to larger dataset, such as the FMA dataset (`https://archive.ics.uci.edu/ml/datasets/FMA%3A+A+Dataset+For+Music+Analysis`, 0.1 million songs, with audio or pre-extracted features), and the Million Song Dataset (`http://millionsongdataset.com/pages/getting-dataset/`, 1 million songs, with features and metadata).

12 Team Contributions

We are three member team, Scott worked on the data extraction part, prepared all the features with necessary plots, RNN; Rongbin worked on the benchmark classification implementation with the sklearn framework and CNN; Huaping worked on the overall methodology design on the benchmark on classical models and paper write up. All the members joined in the results analysis and future work discussion and write-up collaborations.

12.1 Acknowledges

We really appreciated all the help of the professor Tengyu Ma and Christoher Re, and all the TAs.

References

[1] Mingwen Dong. Convolutional neural network achieves human-level accuracy in music genre classification. *CoRR*, abs/1802.09697, 2018.

[2] Y. Panagakis, C. Kotropoulos, and G. R. Arce. Music genre classification via sparse representations of auditory temporal modulations. In *2009 17th European Signal Processing Conference*, pages 1–5, Aug 2009.

[3] George Tzanetakis and Perry R. Cook. Musical genre classification of audio signals. *IEEE Trans. Speech and Audio Processing*, 10:293–302, 2002.

[4] Honglak Lee, Peter T. Pham, Yan Largman, and Andrew Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS*, 2009.

[5] Sag – mark schmidt, nicolas le roux, and francis bach, minimizing finite sums with the stochastic average gradient.

[6] Saga: A fast incremental gradient method with support for non-strongly convex composite.

[7] Limited-memory bfgs.

[8] Chih-Jen Lin (2011). Hsiang-Fu Yu, Fang-Lan Huang. Dual coordinate descent methods for logistic regression and maximum entropy models.

[9] cs229 final project github code.