Machine Learning Algorithms for Music Genre Classification

Konstantinos Kalais
University of Thessaly, Volos
kkalais@inf.uth.gr

https://github.com/Kkalais/Music_Genre_Classification

Abstract

Machine Learning techniques have proved to be able to identify trends from large pools of data, and ultimately classify the music. In that way we will describe classification methods that recognize the genres of music, after analysing tracks metadata, using supervised learning techniques.

1. Introduction

1.1. The problem

Like never before, the web has become a place for sharing creative work - such as music - among a global community of artists and art lovers. While music and music collections predate the web, the web enabled much larger scale collections.

Whereas people used to own a handful of vinyls or CDs, they nowadays have instant access to the whole of published musical content via online platforms. Such dramatic increase in the size of music collections created the need to automatically organize a collection of music samples, as users and publishers cannot manage them manually anymore.

University of Thessaly.

1.2. The motivation

Genre classification is an important task with many real world applications. A 2018 number suggests that ten thousands songs were released every month on internet platforms such as Soundcloud and Spotify. So as the quantity of music being released on a daily basis continues to skyrocket, the need for accurate metadata required for database management and search/storage purposes is increased.

Being able to instantly classify songs in any given playlist or library by genre is an important functionality for any music streaming/purchasing service, and the capacity for statistical analysis that correct and complete labeling of music and audio provides is essentially limited.

Also, successful implementation of genre classification can lead to more personalized music recommendations and well-defined music generation systems.

1.3. Related Works

Machine learning techniques for music genre classification have been studied by many.

In 2018, Hareesh Bahuleyan [1] utilizes hand-extracted features of an Audio dataset's samples and trained four traditional machine learning classifiers (namely Logistic Regression, Random

Forests, Gradient Boosting and Support Vector Machines) with 65% accuracy.

Danny Diekroeger [2] tried to classify songs based on the lyrics using a Naive Bayes classifier, but concluded that analyzing solely lyrics is not enough.

In this study, we will build on top of the works done before and see if we can improve them by applying classical algorithms and neural networks on a feature selected dataset.

2. Dataset Construction and Preprocessing

Experiments were carried out on a dataset for music analysis called FMA [3] (Free Music Archive), an open and easily accessible dataset suitable for evaluating several tasks in Music Information Retrieval, a field concerned with browsing, searching, and organizing large music collections. FMA provides 106.574 music pieces - with pre-computed features and track level metadata - categorized in 16 musical genres, which we split them into training and testing set with 30% test size.

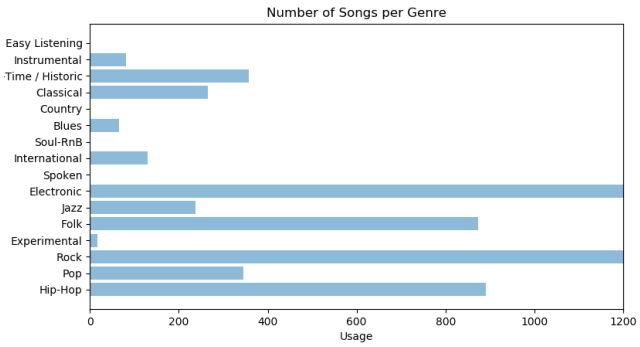


Figure 1. Number of songs per genre

The dataset had some minor inconsistencies such as missing features that we removed during the dataset cleaning phase. It includes 252 features and after dropping the highly correlated ones - with at least 80% correlation, in order to make our algorithm more cost and time efficient - it consists of 24 features including audio features such as instrumentalness, acousticness, energy, tempo, danceability etc.

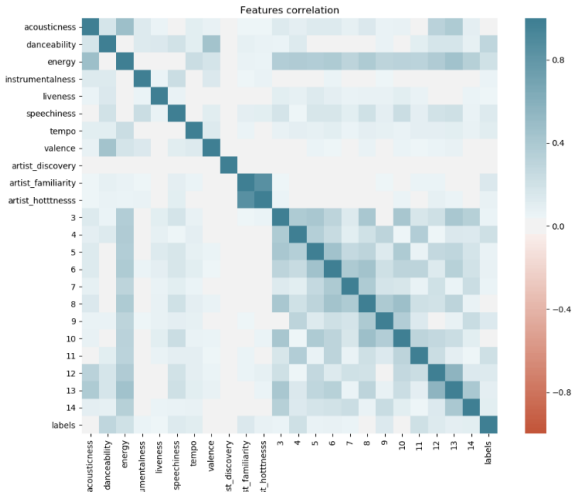


Figure 2. Plot of features correlation

Feature	Description
Instrumentalness	The amount of vocals in the song
Acousticness	How acoustic a song is.
Liveness	The probability that the song was recorded live
Speechiness	The presence of spoken words in a track
Energy	A perceptual measure of intensity and activity.
Dancebility	How suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, and beat strength
Valence	The musical positiveness conveyed by a track. Tracks with high valence sound more positive (happy, cheerful, euphoric), while tracks with low valence sound more negative (sad, depressed, angry)
Tempo	The overall estimated tempo of a track in beats per minute (BPM).

Figure 3. FMA basic features

Next, we use the LabelEncoder class provided by Scikit Learn library, in order to transform the nominal values to numeric and enable the algorithm to perform arithmetic operations on these values.

Lastly, we scaled our data using the MinMaxScaler class of Scikit Learn in the range of (0,1) for easier convergence. This procedure was not implemented for the SVM methodology, as SVMs are scale invariant.

3. Classification Methods

3.1. General Approach and Algorithms used

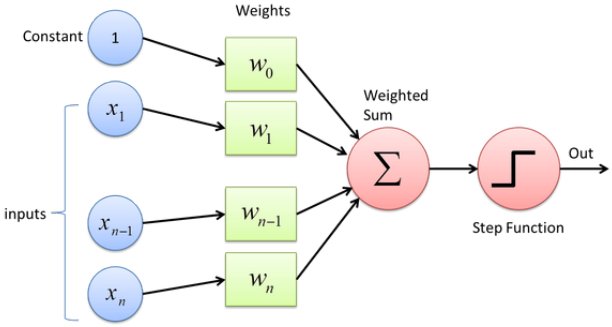
Based on the papers we mentioned, we decided on the use of four algorithms to focus on the issue:

- Multilayer Perceptron
- Random Forest
- Support Vector Machine
- Gradient Boosting

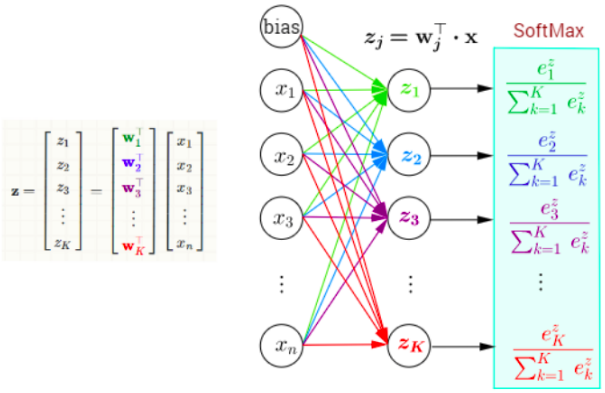
We will analyse below the work implemented in each one.

3.2. Multilayer Perceptron

Neural networks are built off the model of neurons present in the human brain. Neural networks are typically stacked layers of interconnected neurons or nodes each having an activation function. The idea is to give our network several inputs (each input is a single feature). Then, we navigate through the network by applying weights to our inputs, summing them, and finally using an activation function to obtain the subsequent nodes. This process is repeated multiple times through the hidden layers, before applying a final activation function to obtain the output layer, our prediction. An example with n input nodes and 1 output node is shown below, with the step activation function.



The training takes place on the weights, which are initialized randomly. They are updated by trying to minimize training error using gradient descent, and are used after the training period to try and make predictions. In our case, we construct a probability distribution of the 14 genres by running the outputs through a softmax activation function.



Defining the softmax as

$$\sigma(j) = \frac{\exp(\mathbf{w}_j^\top \mathbf{x})}{\sum_{k=1}^K \exp(\mathbf{w}_k^\top \mathbf{x})} = \frac{\exp(z_j)}{\sum_{k=1}^K \exp(z_k)}$$

The model was compiled with Adam optimizer, since its is appropriate for problems with very noisy/or sparse gradients, and a categorical cross-entropy loss function, since we have multiple classes, and trained in batches of 150 instances for 50 iterations.

For the fully connected neural networks architecture we used:

- A Dense input layer of 150 neurons and rectifier linear unit activation function

$$\mathrm{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0. \end{cases}$$

- A Dense hidden layer of 100 neurons and rectifier linear unit activation function.

- A Dense output layer of 16 output classes and softmax activation function.

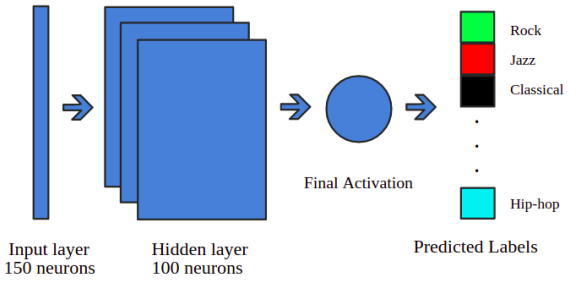


Figure 4. Neural Network

```
model = Sequential()
model.add(Dense(150, input_dim=train_x.shape[1], activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['mse','accuracy'])
model.fit(train_x, train_y, epochs= 50, batch_size=150 , shuffle=False, verbose = 1)
```

Figure 5. MLP model using Keras, Tensorflows high-level API

3.3. Support Vector Machine

SVMs transforms the input feature space into higher-dimensional feature space using the kernel trick dot product, where (x,y) is replaced with k(x,y) = <f(x),f(y)> (f is called kernel function). The transformed data can be separated using a hyperplane, the dividing curve between distinct classes. The optimal hyperplane maximizes the margin, the distance from a data point to the hyperplane.

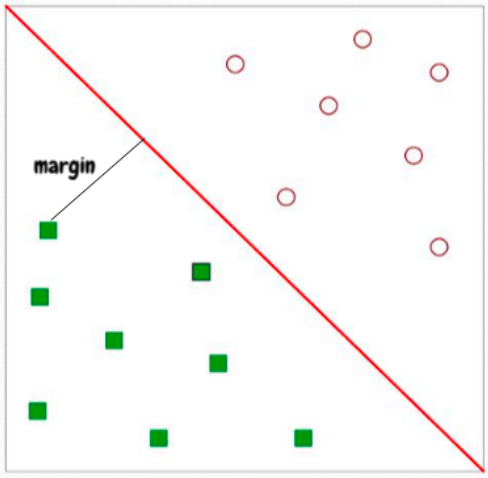


Figure 6. Hyperplane in 2d feature space

In our problem, a radial basis function (RBF) kernel is used to train the SVM because such a kernel would be required to address this non-linear problem.

$$k(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right)$$

Figure 7. Gaussian (RBF) kernel equation

The best parameters we used in the Support Vector Machine technique, after testing several variants with trial-and-error approach, were 10 for the C value, and 1 for the gamma value.

```
C = 10
gamma = 1
classifier = SVC(kernel="rbf",C=C,gamma=gamma)
classifier.fit(train_x, train_y)
y_pred = classifier.predict(test_x)
```

Figure 8. SVM using Sklearns SVC classifier class

3.4. Random Forest

Random Forest is an ensemble of Decision Trees. The general idea of this technique is that a combination of learning models increases the overall result. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. In that way, it prevents overfitting by creating random subsets of the features, building smaller trees using these subsets and combines these subtrees.

As we can see in the figure below, a simplified Random Forest could categorize a sample to the class with the maximum ”votes” among each subtree.

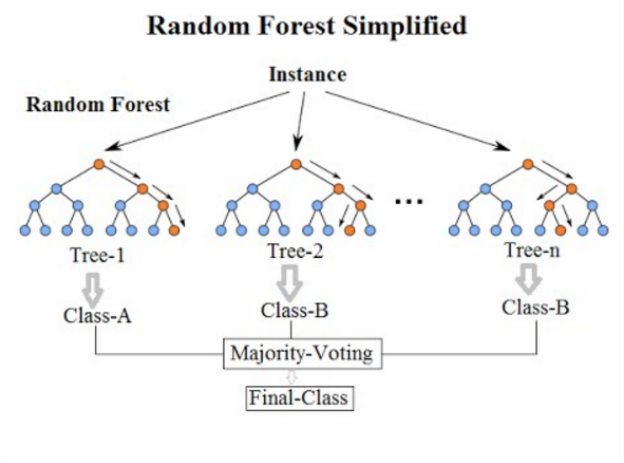


Figure 9. Random Forest Simplified

Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

The parameters we used after trying several variants, were 29 estimators and 29 as the depth of the tree. The algorithm provided us with a relatively high accuracy result in a short time, specifically 68.74% in 1.29 seconds. Generally our opinion was that the algorithm provided us with the best results in regard to how simple it was to develop and fast to execute.

```
clf = RandomForestClassifier(n_estimators=29, max_depth=29,random_state=1)
clf.fit(train_x,train_y)
y_pred = clf.predict(test_x)
```

Figure 10. RF using Sklearns RF classifier class

3.5. Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. In boosting, each new tree is a fit on a modified version of the original data set. The GB begins by training a decision tree in which each observation is assigned an equal weight. After evaluating the first tree, we increase the weights of those observations that are difficult to classify and lower the weights for those that are easy to classify. The second tree is grown on this weighted data. Here, the idea is to improve the predictions of the first tree. Our new model is Tree 1 + Tree 2. Then we compute the classification error from this new 2-tree ensemble model and grow a third tree to predict the revised residuals. We repeat this process for a specified number of iterations. Subsequent trees help us to classify observations that are not well classified by the previous trees. Predictions of the final ensemble model is the weighted sum of the predictions made by the previous tree models.

After trying several variants, we set the values 0.1 and 150 to the learning rate and the estimators - the number of boosting stages to perform - respectively and train our model with 74.21% accuracy, the highest between the four algorithms. Though, Gradient Boosting takes 39.77 seconds to execute, much slower than the above-mentioned techniques.

```
clf = GradientBoostingClassifier(learning_rate=0.1,n_estimators = 150,verbose=1)
clf.fit(train_x,train_y)
y_pred = clf.predict(test_x)
```

Figure 11. GB using Sklearns GB classifier class

3.6. Performance Comparison / Results

The quantitative metric which we used to judge our models is the combination of accuracy (that is, percentage of predicted labels which matched their true labels) and execution time. Below, we provide an (accuracy,time)-based comparison between the four algorithms in order to give a better look into which algorithm is considered to be more suitable.

From the table, it is obvious that Random Forest produces the better results combining the metrics of time and accuracy, and we also provide a bar plot to visually confirm that Random Forest performed better.

Table 1. Classification accuracies and execution times

ALGORITHM	ACCURACY	EXECUTION TIME
GB	74.21%	39.77s
MLP	65.14%	11.21s
SVM	66.74%	3.97s
RF	68.74%	1.29s

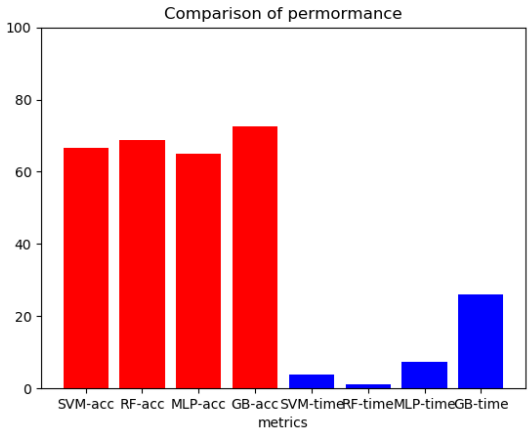


Figure 12. Performance comparison of the four algorithms

Another way of visualizing the performance of our best model is through the confusion matrix as seen in the figure below. We see that GB did well with the rock, jazz and instrumental genre.

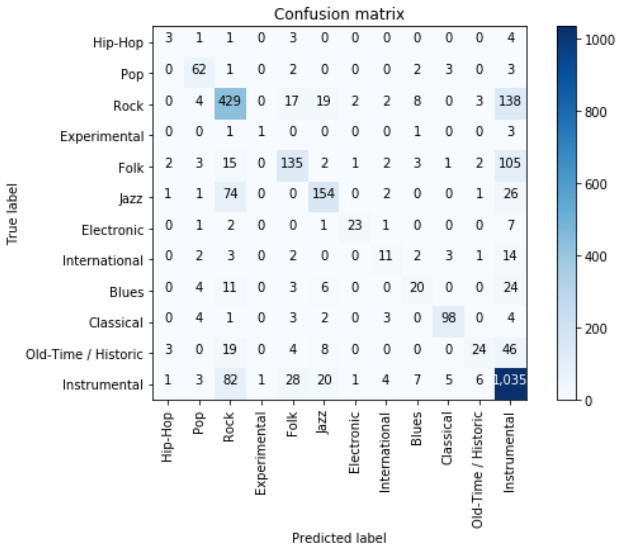


Figure 13. Confusion matrix for GB predictions

Additionally, it incorrectly classified some rock, as well as folk, as instrumental music. We should expect these misclassifications since rock is a genre that both encapsulates many different styles (hard rock, progressive rock, indie rock, alternative rock, etc.) and heavily influences many other derivative genres. Finally, only few samples can be labeled as Hip-hop or Experimental music - classes with the least importance - so it would not make any difference if we had not included them in our dataset.

4. Future work

There are several directions that would be interesting to pursue in the future:

- Experiment with other types of deep learning methods - for example, Convolutional Neural Networks - that they will produce higher classification accuracies
- Try classifying music by another target-feature, e.g. by artist or by decade
- If we are able to train a much deeper network that attains high accuracy, it would be interesting to plot the accuracy of a classical algorithm using different activation layers.

5. Acknowledgments

The project was implemented in the spectrum of a special issue course of the University of Thessaly, Electronic and Computer Engineering Department. The professor and supervisor of the project, prof. Yota Tsompanopoulou guided us to the majority of the knowledge, tools and skills needed for this purpose.

References

[1] Hareesh Bahuleyan, *Music Genre Classification using Machine Learning Techniques*, University of Waterloo, ON, Canada, 2018

[2] Danny Diekreoger, *Can Song Lyrics Predict*

Genre?, Stanford University, 2012

[3] Michal Defferrardy, Kirell Benziy, Pierre Vandergheynsty, Xavier Bresson. `Fma: a dataset for music analysis`, LTS2, EPFL, Switzerland / SCSE, NTU, Singapore, 2017