UNIVERSITY OF PORTO – SCHOOL OF SCIENCES – DEPARTMENT OF COMPUTER SCIENCE

AI approach for classification of musical genre of audio samples

Analysis and comparison of three classification algorithms

Duarte Pereira, 201704138
Isac Novo, 200403278
Rui Ramos, 201705782
March 30, 2021

1 INTRODUCTION

Music has always been both multi-faceted and deeply rooted in each cultural background. People all around the world have been making music long before the first orchestras ever appeared, and different ethnological and epochal influences contribute to produce completely unique and distinguishable musical qualities - classified as genres.

This classification, that has been evolving over time, is not always easy to achieve. Sometimes a genre can be hard to define, despite music of the same genre sharing similar properties such as tempo, beat, and rhythm.

As such, devising an algorithm to classify music by its genre is a nigh impossible task. An efficient alternative to hard coding such program, is to use machine learning algorithms to accomplish this task. These Artificial Intelligence models are able to learn from data containing already classified music, and attempt to accurately determine the genre of further audio samples.

For this assignment, we developed a Python script to create a data set of extracted musical features, and used that data to train three distinct machine learning algorithms R, for solving this classification task. We then analyzed and compared the accuracy of each of these three algorithms.

2 Background

2.1 Classification Problem

2.2 Machine Learning

Machine learning is an area of study and research that focus on computer algorithms that automatically improve by usage, improving in efficiency and accuracy through the use of data as if learning from and acquiring experience. A subfield of Artificial Intelligence, these algorithms build a model based on sample data - *training data* - and use it to predict or make decisions without being explicitly hard-coded to do so.

Although not its only application field, some of the broader uses of machine learning are related to applications for which it is unfeasible to develop conventional algorithms for perform the task. Image recognition or message filtering are examples of such tasks, and so is identifying aspects of music, as our project is focused on.

A subset of machine learning is closely related to computational statistics, making predictions based off of statistical learning. Therefore, machine learning can sometimes be more data analysis exploratory in nature. In this aspect, it is akin to the data mining field of study, applied to predictive analytics via unsupervised training.

For this assignment, and thanks to our background knowledge of data mining algorithms, we adopted a somewhat hybrid approach. Using a data base of audio samples with clearly defined genres, we first extracted relevant features from those audio files with recourse to the Librosa Python's library, in order to produce a data set for supervised training. Then, we proceeded to feed that data set to three distinct predictive models in R - K-nearest neighbors, naive Bayes, and Random Forest.

Finally, we measured each model's efficiency in recognizing the musical genres of further audio samples, given the extracted features.

2.3 K-Nearest Neighbors

The k-nearest neighbors [7] (kNN) algorithm can be both used for classification as well as regression. For our purposes, we obviously wanted the classification output, where an object's class membership is defined by a plurality vote from its nearest neighbors. The object, in this case a audio sample, is assigned a class from the most common one amongst its nearest k neighbors, with k being a positive integer.

In this way, the function of this classification type is a local approximation, allowing for weighted neighboring contributions in reverse to their distance. Training examples are already classified vectors in multidimensional space, according to their features. A simple representation of how this algorithm works, is by positioning the element to be classified along this space, and classifying it in relation to its nearest already classified examples. This distance metric can be computed in several ways, such as the Euclidian distance for continuous variables, of text classification for discrete ones.

Adjusting the value of k, and thus the classification sensibility, the algorithm can learn to improve. Nevertheless, in data sets where class distribution is skewed, a more prevalent class

can dominate the prediction of the new example. to counter this, applying weight to the classification taking into account the distance to each nearest neighbor.

2.4 Naive Bayes

The Byes' theorem describes the probability of an event based on previous knowledge of its related conditions, even if not related to the event. It can be applied to classification of an element, assuming it is representative of a large set; for example, in assessing an individual's health deterioration based on its age alone, given the demographic for a population's overall health according to age.

Naive Bayes [12], are therefore classifiers based on this theorem, and with strong independence assumptions between the considered features - the naiveness, in this context.

The implementation technique revolves around constructing simple classifiers, which are models with assigned class labels for problem instances. These classifiers are represented as vectors of feature values, and class labels are drawn from a finite set. A family of classifier training algorithms exist, that follow the same common principle. That is, and as the name so suggests, all naive Bayes classifiers assume that all feature values are independent of each other, and no correlation is assumed or derived between then, for a given class variable.

2.5 Random Forest

A random forest [11], analogous to a literal forest, can be though of as a series of trees; in this instance, decision trees. A decision tree is a tree-like predictive model containing decisions and their outcomes - including probabilistic ones - along resource costs and utility. It serves as a decision support mechanism, and can be used to display algorithms containing only conditional control statements.

For the a classification problem, a decision tree extracts a class for an object depending on its observable attributes, following the conditional statements of each node. Nodes (or leaves) of the tree represent class labels, while edges (or branches) are defined as conjunctions of features that lead to those class labels.

Random forests employs a multitude of decision trees, built during the training stage of the algorithm. The random forest classification is then the mode of all the classes outputted by each decision tree. This trait allows random forest to surpass the overfitting tendency of a single decision tree and usually outperform them.

2.6 Music Information Retrieval

By considered musical genres as the target classes for our three machine learning models, quantifiable attributes extracted from each audio file are needed to use as the attributes on both the training and the actual classification data sets.

Retrieving information from music, as needed for the classification problem, is the focus of the interdisciplinary science that is Music information retrieval [4] (MIR). Extracting properties from the sound wave or other attributes of an audio file, aside from the obvious parameters like frequency or amplitude, can enable the derivation of several other musical metrics such as rhythm, tempo, and beat.

This approach has proved more fruitful to the classification task than merely scanning a musical sheet, since musical scores are rarely available for commercial or live music and, when so, usually not for every instrument.

Nevertheless, consideration of the audio sample format is needed. The MIDI format, which might appear to be the best format to work with - due to its small size and ease of manipulation - actually entails considerable data loss upon conversion. In that sense, it is not recommended as data source, unless the music was directly written in MIDI.

The popular mp3 or ogg format can be used but, despite little or no difference to the human ear, crucial data for analysis may be missing due to them being lossy formats. That is to say, to reduce file size, these formats compress audio with inexact approximations, partially discarding some of the data.

WAV, also a very popular audio format, is uncompressed raw audio. As such, it constitutes a good format to work with for accurately measuring our machine learning models. Hence, classification efficiency is purely dependent on the model used and the attributes being extracted, rather than being affected by fortuitous poor audio fidelity.

3 METHODS

In this section we are going to explain the various methods used to be able to classify the genre of a given song. We will cover everything from how it was necessary to divide each song into a set of attributes, to finally demonstrate the various algorithms used to classify that song given as a input test.

3.1 FEATURE EXTRACTION

In order to classify audio files by genre, we need to determine which features from the audio itself might be relevant to accomplish the task. As mentioned on the previous section, feature extraction is the process by which meaningful data is obtained from the audio file.

We considered five different features from each audio sample, which we measured and outputted to an CSV file with our Python program. The five extracted features, and a summary of how their extraction was accomplished is summarized below:

- **Root Mean Square (RMS)**

  The root mean square is defined a the square root of the arithmetic mean of the squares of a set of numbers [6]. It is a quadratic mean that can be defined for a continuously varying function in terms of an integral of the squares of instantaneous values during a cycle.

  For this context, it is useful for representing the average power output over a prolonged period of time. Specifically the loudness of a waveform, taking into account all instances of loudness statistics in that waveform.

- **Zero-Crossing Rate (ZCR)**

Zero-crossing refers to a point where a the sign of a mathematical function changes - either from a positive value to a negative one, or vice-versa [10]. This transition can be easily identified in a function's graph by an interception with the axis.

The ZCR, therefore, refers to the rate at which the signal change occurs. This metric has been widely employed in both speech recognition and music classification.

- **Spectral Centroid**

  One of the measures used to characterize a spectrum, the Spectral Centroid indicates where its center of mass is located [5].

  Not to be confused with the median of the spectrum, the Centroid is calculated as the weighted mean of the frequencies present in the signal, determine using a Fourier transform, with their magnitudes as the weights.

  It has been associated, in terms of sound timbre, with its brightness. Which, analogous with visual brightness, it is acoustically formalized as the amount of high-frequency content in a sound.

  For this reason it extensively used in digital audio and music processing as an automatic measure of musical timbre.

- **Spectral Roll-off**

  This feature is the measure of the spectrum's frequency value under which a given percentage of the signal - designated as cutoff - of the total energy of the spectrum is contained [8].

  It can be used to distinguish between harmonic or noisy sounds, whether a frequency is below or above the roll-off value, respectively.

- **Mel-Frequency Cepstral Coefficients**

  The Mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound [2], based on linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

  These coefficients concisely describe the overall shape of a spectral envelope, modeling the characteristics of the human voice. As such, it can applied to speech recognition tasks such as recognizing human spoken words. It also has uses for genre classification given similarities on measures of human vocals for certain musical styles.

- **Chroma Frequencies**

  A chroma feature or chromagram relates to the twelve different pitch classes found in western music [1]. Also referred to as pitch class profiles, these features allow to analyze and categorize music by capturing its harmonic and melodic characteristics, while still being robust to changes in timbre and instrumentation.

3.2 Python Implementation

The reason for choosing Python as the programming language to code our feature extraction program was, as briefly mentioned before, mainly due to the existence of the Python Librosa library [3]. This package provides audio and music analysis tools, and is specifically designed to create music information retrieval mechanisms.

For our script to work, several packages need to be imported, which comprised libraries with useful functions to abstract and simplify our code.

These packages, and their summarized functions, are described below. The import code can be seen on Listing 1.

- **librosa**: in order to extract desired features from songs;
- **numpy**: for arithmetic computation along the specified variable;
- **os**: for file operation functionalities.
- **csv**: to format and write the output in a CSV file.

Listing 1: Necessary packages

```
1  # feature extractoring and preprocessing data
2  import librosa
3  import numpy as np
4  import os
5  import csv
```

Listing 2: CSV header

```
1  #csv header with file name and all attributes to classification
2  header = 'filename␣chroma_stft␣rms␣spectral_centroid'+
3      'spectral_bandwidth␣rolloff␣zero_crossing_rate'
4  for i in range(1, 21):
5      header += f'␣mfcc{i}'
6  header += '␣label'
7  header = header.split()
8
9  #write header into csv
10 file = open('data.csv', 'w', newline='')
11 with file:
12     writer = csv.writer(file)
13     writer.writerow(header)
```

The next step is to create the CSV header with all the important columns for the attributes, as shown Listing 2. This has to be done before extracting the features and assigning them their appropriate columns, one table row per audio file.

For each song in each genre, the values for the relevant features where extracted with the Librosa package, and added to the table as they were being computed. Listing 3 contains the code written for extracting the audio features and writing their values to the CSV file.

Table 3.1 displays the general layout of the table represented by the thus generated CSV file. The first column (aside from the row number) is the name of the audio file, and the last is a label with its appropriate (actual) musical genre. The remaining columns, between those two, each define one of the extracted features and accommodate its values.

Listing 3: Write to CSV

```
#genres
genres = 'blues␣classical␣country␣disco␣hiphop␣jazz␣metal␣pop␣reggae␣rock'.split()

#write each attribute of each song in data.csv
for g in genres:
    for filename in os.listdir(f'genres/{g}'):
        songname = f'genres/{g}/{filename}'
        # Load the audio as a waveform y
        # Store the sampling rate as sr
        y, sr = librosa.load(songname, mono=True, duration=30)
        chroma_stft = librosa.feature.chroma_stft(y, sr)
        rms = librosa.feature.rms(y)
        spec_cent = librosa.feature.spectral_centroid(y, sr)
        spec_bw = librosa.feature.spectral_bandwidth(y, sr)
        rolloff = librosa.feature.spectral_rolloff(y, sr)
        zcr = librosa.feature.zero_crossing_rate(y)
        mfcc = librosa.feature.mfcc(y, sr)
        to_append = f'{filename}␣{np.mean(chroma_stft)}␣{np.mean(rms)}'
        to_append += f'{np.mean(spec_cent)}␣{np.mean(spec_bw)}␣{np.mean(rolloff)}'
        to_append += f'{np.mean(zcr)}'
        for e in mfcc:
            to_append += f'␣{np.mean(e)}'
        to_append += f'␣{g}'
        #append attributes to csv file
        file = open('data.csv', 'a', newline='')
        with file:
            writer = csv.writer(file)
            writer.writerow(to_append.split())
```

1	filename	chroma-stft	rms	...	mfcc19	mfcc20	label
2	blues.00088.wav	0.377977	0.142049	...	-2.535688	0.128512	blues
3	blues.00011.wav	0.367137	0.065713	...	1.836151	-4.897420	blues
4	blues.00074.wav	0.367157	0.196956	...	-3.696436	-1.965981	blues
5	blues.00028.wav	0.275974	0.099696	...	-0.932405	-2.699273	blues
6	blues.00084.wav	0.396258	0.235238	...	-4.380430	0.414055	blues
...	...	...	...	...	...	...	...
997	rock.00071.wav	0.454292	0.160989	...	-5.099720	2.622137	rock
998	rock.00044.wav	0.311549	0.144370	...	3.158420	2.257767	rock
999	rock.00061.wav	0.299721	0.088379	...	1.697010	-1.807093	rock
1000	rock.00009.wav	0.376627	0.128266	...	3.852556	3.555840	rock
1001	rock.00086.wav	0.390935	0.076317	...	-4.932690	-2.615138	rock

Table 3.1: Schematic of the table represented by the CSV file

3.3 Supervised Training Task

After exporting the CSV file with the attributes of 1000 songs, classified into 10 different genres, we set out to train the classification algorithms. These machine learning models were implemented using the R programming language.

The initial data set was divided into two separate sets. The first one, containing 70% of the initial set's size, was used as the training set for the supervised training step. The remaining 30%, was left for the classification task, in order to analyze each of the used machine learning model's efficiency.

Listing 4, shows the code written for partitioning the data set into these other two, as described. While Listing 5, shows the code for both the training and classification steps for each of the three models used. Namely, and as previously stated; kNN, naive Bayes, and random forest.

Listing 4: Data set partition

```
# Import dataset
dataset <- read.csv("data.csv", header=TRUE)
# remove file name column
dataset <- dataset[,-1]

# separate the data set into 70% / 30% for training and test set
idxs <- sample(1:nrow(dataset),as.integer(0.7*nrow(iris)))
train_dataset <- dataset[-idxs,]
test_dataset <- dataset[idxs,]
train_dataset <- as_tibble(train_dataset)
test_dataset <- as_tibble(test_dataset)
```

Listing 5: Classification Algorithms

```
# knn
knn.model <- knn3(label ~., data=train_dataset, k=50)
knn.preds <- predict(knn.model, test_dataset,type="class")

# naive bayes
nb.model <- naive_bayes(label ~., data=train_dataset)
nb.preds <- predict(nb.model, test_dataset,type="class", laplace=1)

# random forest
randomForestModel <- randomForest(label ~ ., data = train_dataset, ntree = 5000, mtry =
predValid <- predict(randomForestModel, test_dataset, type = "class")
```

4 Results and Discussion

For our initial analysis, we ran each algorithm 10 times, in an experimental task to note the first results. For this, we used confusion matrices to obtain the accuracy of each model. Relevant R code can be seen on Listing 6.

The results of applying the confusion matrix to determine each algorithm's accuracy, are present on Table 4.1.

Listing 6: Generation of confusion matrices.

```
# knn
knn.model <- knn3(label ~., data=train_dataset, k=50)
knn.preds <- predict(knn.model, test_dataset,type="class")
knn.confM <- confusionMatrix(test_musical_genres$value, knn.preds)
knn.confM # accuracy

# naive bayes
nb.model <- naive_bayes(label ~., data=train_dataset)
nb.preds <- predict(nb.model, test_dataset,type="class", laplace=1)
nb.confM <- confusionMatrix(test_musical_genres$value, nb.preds)
nb.confM # accuracy

# random forest
randomForestModel <- randomForest(label ~ ., data = train_dataset,
    ntree = 5000, mtry = 6, importance = TRUE)
randomForestModel # accuracy
```

Experience	kNN	Naive Bayes	Random forest
1	28.57%	37.14%	65.92%
2	27.62%	37.14%	65.7%
3	31.43%	37.14%	66.26%
4	29.52%	37.14%	65.47%
5	27.62%	37.14%	65.36%
6	29.52%	37.14%	66.03%
7	31.43%	37.14%	66.03%
8	28.57%	37.14%	66.37%
9	27.62%	37.14%	65.81%
10	28.57%	37.14%	65.92%

Table 4.1: Algorithms Accuracy.

As expected [9], by comparing the results obtained, the random forest algorithm outperformed the other two. Notably better classifications can be achieved with the random forest, given the extracted features and the training set.

Of the other two, the naive Bayes' algorithm produced better classifications than the kNN. This is somewhat surprising, and serves to demonstrate that, in terms of musical genre classification, independent classification produces better results than the ones based on a proximity metric.

To verify these results empirically, we classified 23 songs of different genres with the created models. The results are depicted on Table 4.2.

The first thing to note is that when the three algorithms produce the same classification, that genre is the correct one. Classical music is easily recognizable from the rest, according to this behaviour.

Still, only the random forest model was able to differentiate classical music from jazz or blues. Despite this, it failed to correctly identify a blues song in one instance.

Further expanding on this observation, it is also possible to ascertain that results vary for different songs within the same genre. For example, A metal song from System of a Down and another from Slipknot were wrongly classified by two of the three models, with those same models getting a correct classification on a third instance.

These hints to the fact that these results may be more accurate if musical sub-genres are taken into consideration. Since music from different bands, even sometimes from the same band, can't always be classified with a blanket musical genre.

Experience	Real Genre	kNN	Naive Bayes	Random forest
Test Music 1	Blues	Classical	Country	Blues
Test Music 2	Classical	Classical	Classical	Classical
Test Music 3	Rock	Disco	HipHop	Rock
Test Music 4	Disco	Metal	Disco	Disco
Test Music 5	HipHop	Blues	Reggae	HipHop
Test Music 6	Reggae	Country	Reggae	Reggae
Test Music 7	Pop	Pop	Pop	Pop
Test Music 8	Country	HipHop	Pop	Country
Test Music 9	Jazz	Classical	Classical	Jazz
Test Music 10	Metal	Metal	Metal	Metal
Slipknot	Metal	Metal	HipHop	Disco
Bethoven	Classical	Classical	Classical	Classical
Selena Gomez	Pop	Country	Classical	Country
Ottawan	Disco	Country	Reggae	Rock
Improvisation	Blues	Classical	Classical	Blues
DeathStarts	Metal	Rock	HipHop	HipHop
Blues Delight	Blues	Jazz	Classical	Country
Autumn Leaves	Jazz	Classical	Classical	Jazz
Nirvana	Rock	Country	Reggae	Reggae
Bob Marley	Raggae	Classical	Jazz	Blues
Bethoven	Classical	Classical	Classical	Classical
Taylor Swift	Pop	Country	Blues	Country
System Of a Down	Metal	Metal	Disco	Disco
Hit Percentage		30,4%	26,1%	60,1%

Table 4.2: Comparison between the actual musical genre and the classification results

5 Conclusions

In this assignment we were able to demonstrate how to implement machine learning algorithms for classifying music by its genre.

We were able to extract features from audio files within certain labeled musical genres, and create a data set with that information. We were then able to use that data to train three different machine learning classification algorithms - k-nearest neighbors, naive Bayes, and random forest.

Finally, we analyzed and evaluated each model's accuracy on solving the proposed categorization task.

Of the three models analysed, we concluded that random forests showed the best accuracy in prediction the actual genre of a music from the extracted attributes. Still, no single algorithm presented high accuracy in its classification. Despite this, as shown by the empirical data, the combined effort of the three algorithms seem to be more useful than each of them isolated.

This tells us that, while further work could be done both on the size and quality of the training data set and the extracted features, another layer of machine learning algorithms could be placed on top of this one, using the results from these three, as data to produce a final classification.

References

[1] Chroma toolbox. https://www.audiolabs-erlangen.de/resources/MIR/chromatoolbox. [Online; accessed 30/03/2021].

[2] Hmm-based audio keyword generation. https://web.archive.org/web/20070510193153/http://cemnet.ntu.edu.sg/home/asltchia/publication/AudioAnalysisUnderstanding/Conference/HMM-Based\%20Audio\%20Keyword\%20Generation.pdf. [Online; accessed 30/03/2021].

[3] Librosa documentation. https://librosa.org/doc/latest/index.html. [Online; accessed 30/03/2021].

[4] Notes on music information retrieval. https://musicinformationretrieval.com/. [Online; accessed 30/03/2021].

[5] Spectral features. https://musicinformationretrieval.com/spectral_features.html. [Online; accessed 30/03/2021].

[6] Understanding what does rms stands for in audio: Definition & details. https://www.audiorecording.me/understanding-what-does-rms-stands-for-in-audio-definition-details.html. [Online; accessed 30/03/2021].

[7] Naomi S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. https://ecommons.cornell.edu/bitstream/handle/1813/31637/

BU-1065-MA.pdf;jsessionid=4B9978463B374D1442B5DD091E002F28?sequence=1, 1992. [Online; accessed 30/03/2021].

[8] Min Xu; et al. Rolloff - essentia documentation. https://essentia.upf.edu/reference/streaming_RollOff.html, 2004. [Online; accessed 30/03/2021].

[9] Luke Fahmy. Naive bayes, knn, and random forest comparison. http://fahmy.xyz.s3-website-us-west-2.amazonaws.com/paper.pdf. [Online; accessed 30/03/2021].

[10] Delerue O Gouyon F., Patchet F. On the use of zero-crossing rate for an application of classification of percussive sounds. https://www.semanticscholar.org/paper/ON-THE-USE-OF-ZERO-CROSSING-RATE-FOR-AN-APPLICATION-Gouyon-Pachet/650914f8be2c96ab2f55faec54d3e3876c5b1b69?p2df, 2000. [Online; accessed 30/03/2021].

[11] Tin Kam Ho. Random decision forests. https://web.archive.org/web/20160417030218/http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf, 1995. [Online; accessed 30/03/2021].

[12] Andrew McCallum. Graphical models, lecture2: Bayesian network represention. https://people.cs.umass.edu/~mccallum/courses/gm2011/02-bn-rep.pdf, 2019. [Online; accessed 30/03/2021].