Naive Bayes Genre Classifier

Robu Petru Razvan

November 2025

GitHub Repository:

https://github.com/petru-robu/Genre-Classifier-Bayes

Contents

1 Introduction

Music genre classification is a well-known problem in the field of machine learning.

Currently, there are many ways to approach this task using more advanced ML techniques and models.

This project attempts to solve the problem in the simplest way possible, by using a Multinomial Naive Bayes Classifier.

2 Dataset

The dataset used is the GTZAN Music Genre Dataset. It contains 1,000 song samples, each 30 seconds long, belonging to a total of 10 genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock.

The dataset provides the audio samples categorized by genre, as well as a CSV file with pre-extracted features. These features result from the spectral analysis of the audio signals. Relevant features include:

- Tempo
- Chroma feature [1]
- Root mean square (RMS) [2]
- Spectral centroid [3]
- Mel-frequency cepstrum (MFC) [4]

These features are float values that give information about the analyzed audio files. Since the Bayes Classifier works on probabilities (frequencies/counts), we need a method to discretize these values.

The chosen approach is to split the interval $[min_f, max_f]$ for a feature f into N bins of equal size, where min_f and max_f are the minimum and maximum values of the feature across the entire training dataset. Each feature value is then assigned to a bin. The Bayes Classifier operates on the frequency of these bins.

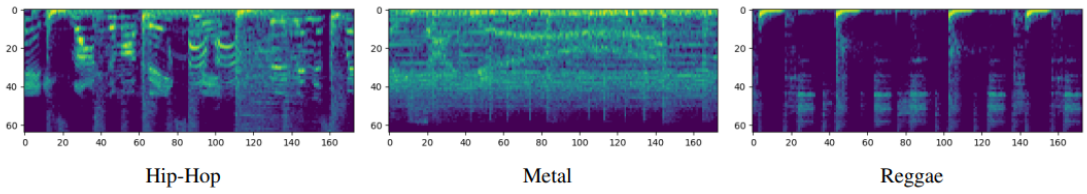

Figure 1: Examples of log-scaled mel-spectrograms for three different genres.

[1] About chroma feature: `https://en.wikipedia.org/wiki/Chroma_feature`
[2] About RMS: `https://en.wikipedia.org/wiki/Root_mean_square`
[3] About spectral centroid: `https://en.wikipedia.org/wiki/Spectral_centroid`
[4] About MFC: `https://en.wikipedia.org/wiki/Mel-frequency_cepstrum`

3 Naive Bayes Classifier

To model our problem, we introduce the following:

- **Classes** - The problem is partitioned into classes, which in our project represent music genres.
- **Features** - Features describe our classes. In this project, features are indicators derived from the spectral analysis of audio files.

The Naive Bayes Classifier assumes that features are independent; this assumption is naive, as features are usually correlated in practice.

For example, a metal song may have a low spectral centroid and a high tempo. Observing a high centroid often implies a high tempo as well.

The Naive Bayes Classifier ignores such dependencies between features.

Bayes' Theorem

Bayes' theorem provides a mathematical rule for inverting conditional probabilities, allowing the probability of a cause to be determined given an effect.

Theorem 1. *Let (C_k) be the classes representing our problem and $\mathrm{F} = (f_1, f_2, \ldots, f_n)$ a vector of features. Then Bayes' Theorem states:*

$$\mathbb{P}(\mathrm{C_k}|\mathrm{F}) = \mathbb{P}(\mathrm{F}|\mathrm{C_k}) \cdot \frac{\mathbb{P}(\mathrm{C_k})}{\mathbb{P}(\mathrm{F})}$$

Where:

- $\mathbb{P}(\mathrm{C_k}|\mathrm{F})$ *is the posterior probability of class C_k given feature F.*
- $\mathbb{P}(\mathrm{F}|\mathrm{C_k})$ *is the likelihood of feature F given class C_k.*
- $\mathbb{P}(\mathrm{C_k})$ *is the prior probability of class C_k.*
- $\mathbb{P}(\mathrm{F})$ *is the marginal probability of feature F.*

In practice, $\mathbb{P}(\mathrm{F})$ can be ignored, as it is constant and independent of the classes.
Predicting a genre involves:

$$\begin{aligned}
\mathbb{P}(\mathrm{C_k}, f_1, f_2, \ldots, f_n) &= \mathbb{P}(f_1, f_2, \ldots, f_n, \mathrm{C_k}) \\
&= \mathbb{P}(f_1 \mid f_2, f_3, \ldots, f_n, \mathrm{C_k})\, \mathbb{P}(f_2, f_3, \ldots, f_n, \mathrm{C_k}) \\
&= \cdots \\
&= \mathbb{P}(\mathrm{C_k}) \prod_{i=1}^{n} \mathbb{P}\big(f_i \mid f_{i+1}, f_{i+2}, \ldots, f_n, \mathrm{C_k}\big)
\end{aligned}$$

Assuming feature independence:

$$\mathbb{P}(\mathrm{C_k}, f_1, f_2, \ldots, f_n) = \mathbb{P}(\mathrm{C_k}) \prod_{i=1}^{n} \mathbb{P}\big(f_i \mid \mathrm{C_k}\big)$$

For implementation, we use log probabilities to avoid overflow:

$$\ln \mathbb{P}(\mathrm{C_k}, f_1, f_2, \ldots, f_n) = \ln \mathbb{P}(\mathrm{C_k}) + \sum_{i=1}^{n} \ln \mathbb{P}\big(f_i \mid \mathrm{C_k}\big)$$

4 Implementation

The Bayes Classifier is implemented in Python using the following libraries:

- numpy - for vectorization and mathematical operations
- librosa - for digital signal processing
- os, json, csv - for data and file management

Structure

The app has three parts:

- **Trainer**, responsible for:
  - Parsing the dataset: Training uses half of the GTZAN dataset, specifically the pre-extracted CSV features to save resources.
  - Computing prior probabilities, $\mathbb{P}(\mathrm{C_k})$.
  - Discretizing features using:

$$d(v, l, r, N) = \begin{cases} 0, & \text{if } v \leq l \\ N-1, & \text{if } v \geq r \\ \min\left(\left\lfloor \dfrac{v-l}{(r-l)/N} \right\rfloor, N-1\right), & \text{otherwise} \end{cases}$$

    with
    - v = input value
    - l = lower bound
    - r = upper bound
    - N = number of bins
  - Computing likelihoods $\mathbb{P}(f_i \mid C_k)$ for each feature bin, including Laplace smoothing:

$$P_i = \frac{c_i + \alpha}{N + \alpha \cdot K}$$

    where:
    - P_i = smoothed probability of bin i
    - c_i = count of occurrences of bin i
    - α = smoothing constant
    - N = total number of observations
    - K = number of bins
  - Saving model priors, feature probabilities, and bounds to a JSON file for the predictor. Note: Trainer is implemented in *trainer.py*; discretization and feature bounds are in *utils.py*.

- **Predictor**, responsible for:
  - Loading the trained model from JSON.
  - Making predictions by computing the log-sum of priors and likelihoods:

$$\ln \mathbb{P}(\mathrm{C_k}, f_1, f_2, \ldots, f_n) = \ln \mathbb{P}(\mathrm{C_k}) + \sum_{i=1}^{n} \ln \mathbb{P}\big(f_i \mid \mathrm{C_k}\big)$$

    Note: Core functionality is in *predictor.py*. Features for prediction can be extracted from audio (*predictaudio.py*) or CSV (*predictcsv.py*).

5 Results

Testing the model on half the dataset yielded a maximum accuracy of 40.07%.

Discretization matters: the 40.07% was achieved with $N_{bins} = 12$. Other results:

- $N_{bins} = 9$: 36.04% accuracy
- $N_{bins} = 40$: 37.88% accuracy
- $N_{bins} = 150$: 34.21% accuracy
- $N_{bins} = 300$: 29.53% accuracy

Reasons for these results:

- Discretizing features loses information. Sparse or overly grouped bins affect model accuracy.
- Genre boundaries overlap. For example, rock, blues, and country may be similar, while metal was identified with 95% probability. Classical matched 68% correctly, 25% as jazz, 7% as other. Feature selection also influenced results.
- Naive Bayes assumes independence. Features from spectral analysis are correlated, reducing accuracy.

6 Resources

- GTZAN dataset: Kaggle GTZAN dataset
- Naive Bayes, GeeksForGeeks: Naive Bayes, GeeksForGeeks
- Naive Bayes, Wikipedia: `https://en.wikipedia.org/wiki/Naive_Bayes_classifier`
- Music Genre Classification, Stanford: `https://cs229.stanford.edu/proj2018/report/21.pdf`