American University of Armenia

Zaven & Sonia Akian College of Science and Engineering

CS 251, Machine Learning Course Project

Classical Music Classification

Team:
Sona Bezirganyan
Hripsime Ghazaryan
Lusine Davtyan

Spring, 2021

Abstract

Classification of music is an essential part of Music Information Retrieval (MIR). The applications include music recommendation systems, music generation, finding of related musical pieces, etc. In this project, we aim to classify pieces of classical music according to their period. Given a piece of classical music, we want to classify it into one of the following categories: Baroque, Classical, Romantic, and Modern. First, the history and the motivation of the problem is discussed, followed by solutions of similar problems. For our project, we chose the MAESTRO dataset, which contains approximately 200 hours of piano compositions. Different models and classification algorithms are used to solve the problem and then compared against each other.

Keywords: Music Information Retrieval, Classification, Classical Music

Contents

Chapter 1

Introduction

1.1 Problem Setting, Project Motivation and Description

The problem lies in classification of classical music pieces according to their periods. Given a piece of classical music, we want to classify it into one of the following categories: Baroque, Classical, Romantic, and Modern.

The roots of music classification lie in the history of the development of Music Information Retrieval (MIR) which is the research area of retrieving data from music and analyzing it. As a term, MIR was first used by Michael Kassler in 1965. MIR has been defined by Stephen Downie as "a multidisciplinary research endeavor that strives to develop innovative content-based searching schemes, novel interfaces, and evolving networked delivery mechanisms in an effort to make the world's vast store of music accessible to all" (Szczepański Szyca, 2014). MIR techniques most commonly label classes based on their genre, mood, instrument or the artist (composer). MIR technologies are used for music classification, music generation and for many other purposes in different fields. Our problem is classical music period classification, so we will label our data in a different way; based on the periods.

J. Marques and P. J. Moreno were one of the first contributors in instrument classification (Marques, Moreno, 1999). The features they used were linear prediction coefficients, FFT based cepstral coefficients, and FFT based mel cepstral coefficients. And the algorithms for training the data were Gaussian Mixture Models and Support Vector Machines. They have experimented with different combinations of features and models and created an eight-instrument classifier. The most successful model in their findings was using 16 mel cepstral coefficients as features together with the SVM classification algorithm. And the inaccuracies sometimes

were assumed to be because of the differences in frequency range in train and test datasets.

Another work of classification of music was done by J. Lebar et al (Lebar, Chang, Yu, 2008), and it is focusing on classification of musical scores according to the composers. To avoid the challenges of audio processing, they used musical scores in the plain-text **kern format. The features were based on the information about articulations, dynamics, note stem directions, and the algorithms used were Naïve Bayes, SVM, Linear and quadratic discriminant analysis and K-Nearest Neighbors. SVM was comparatively more accurate classifier in this problem setting.

Classifying classical music based on the period is relatively less explored. Another work done by Bas van 't Spijker (Spijker, 2020) includes MFCC and Chromagrams as features and use Convolutional Neural Networks, Long Short Term Memory and SVM algorithms. He, also, experimented with all the combinations of the features and the algorithms. Based on the predictions of test sets, the highest accuracy was attained with MFCCs and CNNs.

1.1.1 The Structure of the Project Paper

The project paper consists of 5 sections. The second section is devoted to Data, Preprocessing, and Feature Selection. Methods for measuring the performance of the models are also described in this section. The third section refers to Algorithms and Models used to solve the problem. In the forth section we describe the experiments we have done. This section also includes analysis of results, comparison between different models used, as well as a summary of the project. The fifth section is devoted to the implementation of the algorithms in Python.

Chapter 2

Data and Preprocessing, Performance Measurement

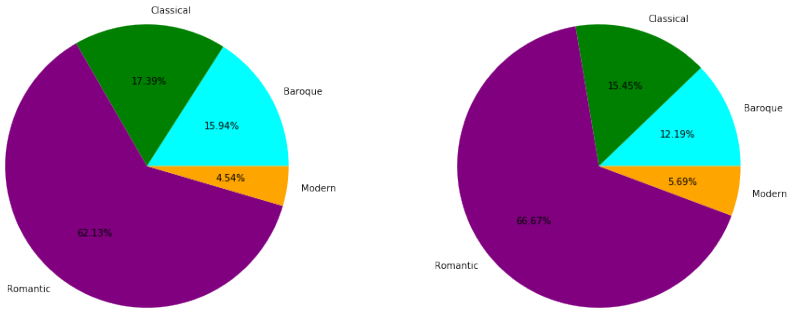
2.1 Dataset

To solve the problem, we use the MAESTRO Dataset (Hawthorne et al., 2019). It contains pieces of classical composers from the 17th to early 20th century. The Dataset consists of over 200 hours of audio and MIDI recordings of piano performances. It includes 1282 pieces from 37 composers. To be applicable to our project, the dataset needed to have labels corresponding to the following 4 periods of classical music: Baroque, Classical, Romantic, and Modern. While there are different approaches of labeling (according to the composer, for instance), often there is no clear distinction between the periods. For example, there are composers whose different works represent different styles. After looking at some approaches to this problem, we decided to use the one that labels the pieces according to the canonically established year of their composition. Thus, we used the labeled version of the MAESTRO dataset which is available here[1] and which sticks to the definition of periods described here[2]. We encoded the labels into integers in the following way:

Label	Integer
Baroque	0
Classical	1
Romantic	2
Modern	3

[1] https://github.com/vantspijker/periodclassification/blob/master/maestro-v2.0.0.csv

[2] https://www.mfiles.co.uk/composer-timelines-classical-periods.htm



(a) Class Distribution of Train Set

(b) Class Distribution of Test Set

Figure 2.1: Class Distribution of Data

2.2 Feature selection and Preprocessing

The related works found during our research mostly use Mel-Frequency Cepstrum Coefficients (MFCCs) and Chroma features. MFCCs are short term representation of the power spectrum of a signal, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. And Chroma features are representing tonal content of the audio and the quality of the pitch class. Another unique approach to a related problem was to classify musical scores in the **kern format. The authors of the project state that they avoided audio processing because of its challenges. That is why they decided to analyze musical scores in **kern format instead. From the information provided by **kern scores, they used pitches, their durations and chords: they have generated features from chords of each score.

Based on our research, we decided to use the `music21` library of Python for extracting features from the MIDI files provided by the MAESTRO dataset. MIDI (Musical Instument Digital Interface) is a protocol (series of messages) allowing digital music tools to communicate. It is making audio processing easier as a few hundred MIDI messages can represent a piece of music. The `music21` is a powerful library of Python for computer-aided musicology. This library was initially developed by MIT and the 21 in library's name comes from the number of courses (including those connected with music) provided by the humanities department at MIT. The library also implements the extractor of features which makes possible extracting features from MIDI files. The features we used include the average note duration, basic pitch histogram, the most common pitch class, pitch class distribution, etc.

We used a top-down approach for feature selection. First, we extracted most of the features that the `music21` library allowed us to. All of the features are numerical, but some of them consist of more than one elements. To get a final feature vector, we treated each element of such vectors as an individual feature, and concatenated all features. At this stage, we had 312 features. Then, we dropped the features that had 0 variance, as well as the ones with a correlation greater than 0.9. After these, the number of features was reduced to 122.

Finally, we have scaled the dataset, since some of the algorithms we use require this to have a goof performance.

The Dataset was already splitted to train/validation/test sets. They consisted of 967, 137, and 178 pieces respectively. However, since we are choosing hyperparameters using the cross-validation technique, we distributed the validation set between the train and test sets.

2.3 Performance Measurement

For each of our models, we have made a full classification record including the accuracy, F1 score, Precision, Recall, etc. The main performance measurement that we have used is the Ordinal Classification Index. It is also the one used to determine the best parameters during cross validation. The description of the measurement as well as the implemerntation are based on the paper by Jaime S. Cardoso and Ricardo Sousa (Cardoso, Sousa, 2011). The reason we consider this index to be an important measurement is that we want to consider the extent of misclassification. For example, if the classification was done incorrectly, it still matters whether our model "confused" Baroque with Classical or with Modern period. There are, also, other performance measurements dealing with ordinal data such as the Spearman's rank correlation coefficient, R_s, and Kendall's τ_b. However, these numbers are only considering the order relation between actual and predicted classes. But we, also, need to incorporate this with the extent of the divergence of the predicted class from the actual one. The Ordinal Classification Index takes into consideration the above mentioned important components. It uses confusion matrix for obtaining that necessary error components. Before going into the details let us give the definition of non-discordant pair of points that we are going to use a lot.

Definition 1 (**Non-Discordant Pairs**). The pair of points x_i and x_j is called *non-discordant* if the relative order of the true classes $\mathcal{C}_{x_i}$ and $\mathcal{C}_{x_j}$ is not opposite to the

relative order of the predicted classes $\hat{\mathcal{C}}_{x_i}$ and $\hat{\mathcal{C}}_{x_j}$ (if there is a tie in either the true or predicted classes, or both, the pair is still *non-discordant*).
In the CM, Definition 1 is equivalent to

$$\text{sign}\left((r_{x_i} - r_{x_j}) \times (c_{x_i} - c_{x_j}\right) \geq 0$$

where r_{x_i} and c_{x_i} are the row and column in the CM corresponding to example x_i, respectively.

Also, let us define what a path and benefit in the confusion matrix are: A path in the confusion matrix is the sequence of entries where two consecutive entries in the path are 8-adjacent neighbors. The benefit of a path is the sum of the values of the entries in the path.

Now we can give the idea of the coefficient we are going to use for measuring the performance of our algorithms. This coefficient allows the pairs to deviate from the main diagonal as long as they stay non-discordant. So, all the pairs forming a consistent path from (1,1) to (K,K) are allowed, where K is the number of classes.

Definition 2. A path is said to be consistent if every pair of nodes in the path is non-discordant.

Besides allowing the pairs to deviate from the main diagonal, we, also, penalize the deviation of our path from the main diagonal. So, we should find the consistent path from (1,1) to (K, K) maximizing the benefit and minimizing the number defining the deviation from the main diagonal. So, the OC index will have the following form:

$$OC_\beta = \min\left\{(1 - \frac{1}{N}\text{benefit(path)}) + \beta(\text{penalty(path)})\right\}$$

Here, we are minimizing over the set of all consistent paths from (1,1) to (K,K) and $\beta \geq 0$. And mathematically rigorous formulation of the OC index is:

$$OC_\beta^\gamma = \min\left\{1 - \frac{\sum_{(r,c)\in\text{path}} n_{r,c}}{N + (\sum_{\forall(r,c)} n_{r,c}|r-c|^\gamma)^{\frac{1}{\gamma}}} + \beta \cdot \sum_{(r,c)\in\text{path}} n_{r,c}|r-c|^\gamma\right\}$$

Where N is the number of observations, $n_{r,c}$ is the number of observations from the r-th class predicted as being from c-th class and $\gamma \geq 1$.

Let us, also, mention some important properties of the Ordinal Classification Index.

OCI satisfies the identity of indiscernible property, i.e., $OC_{\beta}^{\gamma}(a,b) = 0$ if and only if $a = b$.

OC_{β}^{γ} is always non negative since we are considering only consistent paths. And the cost of the path through the main diagonal does not exceed 1. Thus, $0 \leq OC_{\beta}^{\gamma} \leq 1$.

The transposition of the CM does not change the value of OC_{β}^{γ} meaning that $OC_{\beta}^{\gamma}(a,b) = OC_{\beta}^{\gamma}(b,a)$.

The above conditions are important properties of a classification performance index.

Chapter 3

Algorithms and Models

3.1 Algorithms and Models

The models we used to solve our problem are KNN, LDA, QDA, Gaussian Naive Bayes Classifier, SVC and Random Forest. The models were selected because of being able to work with multiclass classification. Moreover, the predictive power of Random Forest is high.

Chapter 4

Results and Conclusion

4.1 Experiments and Results

We have done experiments with our selected models. For KNN, SVC and Random Forest, we used Grid Search to choose the best hyperparameters. As a result, we got the following models.

- `KNeighborsClassifier(n_neighbors=39)`
- `LinearDiscriminantAnalysis()`
- `QuadraticDiscriminantAnalysis()`
- `GaussianNB()`
- `SVC(C=1, gamma=0.01)`
- `RandomForestClassifier(criterion='entropy', max_depth=30, max_features=3, min_samples_leaf=2, random_state=4)`

The tables below show the classification report for each of the models.

K Nearest Neighbors

	precision	recall	f1-score	support
0	0.77	0.67	0.71	30
1	0.50	0.45	0.47	38
2	0.80	0.91	0.85	164
3	0.00	0.00	0.00	14
accuracy			0.76	246
macro avg	0.52	0.51	0.51	246
weighted avg	0.71	0.76	0.73	246

OCI: 0.3230736367873233

LDA

	precision	recall	f1-score	support
0	0.43	0.63	0.51	30
1	0.44	0.53	0.48	38
2	0.79	0.65	0.71	164
3	0.10	0.14	0.11	14
accuracy			0.60	246
macro avg	0.44	0.49	0.46	246
weighted avg	0.65	0.60	0.62	246

OCI: 0.5357204635876146

QDA

	precision	recall	f1-score	support
0	1.00	0.03	0.06	30
1	1.00	0.03	0.05	38
2	0.67	1.00	0.80	164
3	0.00	0.00	0.00	14
accuracy			0.67	246
macro avg	0.67	0.26	0.23	246
weighted avg	0.72	0.67	0.55	246

OCI: 0.42063151265315474

GaussianNB

	precision	recall	f1-score	support
0	0.56	0.30	0.39	30
1	0.39	0.95	0.55	38
2	0.91	0.59	0.71	164
3	0.12	0.29	0.17	14
accuracy			0.59	246
macro avg	0.50	0.53	0.46	246
weighted avg	0.75	0.59	0.62	246

OCI: 0.4402954311233254

SVC

	precision	recall	f1-score	support
0	0.68	0.83	0.75	30
1	0.62	0.34	0.44	38
2	0.82	0.94	0.88	164
3	0.00	0.00	0.00	14
accuracy			0.78	246
macro avg	0.53	0.53	0.52	246
weighted avg	0.72	0.78	0.74	246

OCI: 0.29686887302956644

Random Forest

	precision	recall	f1-score	support
0	0.73	0.80	0.76	30
1	0.72	0.74	0.73	38
2	0.87	0.93	0.90	164
3	0.00	0.00	0.00	14
accuracy			0.83	246
macro avg	0.58	0.62	0.60	246
weighted avg	0.78	0.83	0.80	246

OCI: 0.24731050340806443

Since our main performance measure is the Ordinal Classification Index, let us compare the results using this measure. Below we have ordered the models according to their performance: the better (the lower) the OCI, the higher the position in the table. We can see that the best performance was that of Random Forest's. SVC and KNN are the second and the third respectively. Note that OCI generally agrees with the other performance measures when it comes to ordering the models with respect to their performance. For example, if we ordered the table using the accuracy score, the first 4 positions would remain unchanged, whereas the 5-th and 6-th would get swapped.

	Name	**OCI**
1	Random Forest	0.247311
2	SVC	0.296869
3	KNN	0.323074
4	QDA	0.420632
5	Gaussian NB	0.440295
6	LDA	0.535720

4.2 Conclusion

Our aim was to classify a given work of classical music into one of the following periods: Baroque, Classical, Romantic, and Modern. We used the `music21` library of Python to extract the features, which included the average note duration, basic pitch histogram, the most common pitch class, pitch class distribution, etc. We were able to achieve 83% accuracy using Random Forests. The ordinal classification index, which was used as the primary performance measure was approximately 0.24 for this model. The second and third best results were obtained using SVC and KNN. For SVC, we had 78% accuracy and 0.29 OCI. And for KNN, we got 76% and 0.32 for the accuracy and OCI respectively. One of the drawbacks is the weak performance on observations representing the Modern period. We can see that Random Forests - our best model, did not classify any of the works as modern. This is a result of an imbalanced dataset, since we had a very few number of modern works. Still, there were models, for example Gaussian Naive Bayes and LDA, that did not perform well in terms of accuracy or OCI, but were able to correctly identify some modern works. The best results were achieved with the works from Romantic period. We can come to this conclusion from, say, the F1 score.

	Name	Accuracy
1	Random Forest	0.83
2	SVC	0.78
3	KNN	0.76
4	QDA	0.67
5	LDA	0.60
6	Gaussian NB	0.59

4.3 Further Work

In the future, we want to try to somehow balance our dataset to have a better performance especially on observations having the modern label. Additionally, we want to apply Neural Networks for solving this problem.

Chapter 5

Bibliography

Marques, J., Moreno, P. (1999). A Study of Musical Instrument Classification Using Gaussian Mixture Models and Support Vector Machines. Compaq Computer Corporation.

Lebar, J., Chang, G., Yu, D. (2008). Classifying Musical Scores by Composer: A machine learning approach

Spijker, B. V. (2020). Classifying Classical Piano Music Into Time Period Using Machine Learning.

Szczepański, S. J., Szyca, M. S. (2014). Music Information Retrieval.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C. A., Dieleman, S., ... Eck, D. (2019). Enabling Factorized Piano Music Modeling and Generation with the Maestro Dataset.

Cardoso, J., Sousa, R. (2011). Measuring The Performance of Ordinal Classification.