Comparison of learning models for music genre classification

Abstract
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Keywords
Keyword1 — Keyword2 — Keyword3

1
2

Contents

1. Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus.

Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

2. Data Analysis

Before diving deep into applying some machine learning algorithms it is useful to explore the data a bit to get a general idea of the features and quirks of the data set. Our data analysis process began by printing the values for some randomly selected data points. The raw data does not make much sense and it is challenging to try to find some correlation between certain classes and some selected features. However, in that small sample a disproportionate share of the labels were *Pop_Rock* or class 1. The next logical step was to take a closer look at the class distribution of the given data and see if the observed unbalanced distribution is just randomness or a feature of the data set. The class distribution has been visualized in figure 1 and the dominance of class 1 can be clearly seen. This imbalance might be due to having two popular music generes, pop and rock, under one label but it can also be just a feature of the data set. Intuitively the disproportionate class distribution makes sense but for practical applications that have to work on any unlabeled data set, any assumptions can hardly be made.

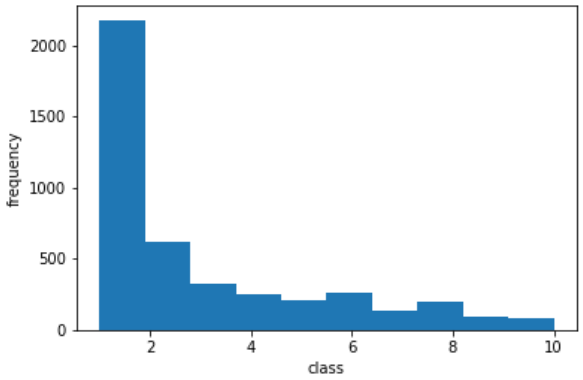


Figure 1. Distribution of classes in the data set

Another interesting feature of the data set is the low number of samples with relation to the amount of features. Since there are 4363 songs and each of them have 264 features, it is obvious that the dataset is nowhere near sufficient to explore all combinations of all features. Most likely the whole space is not needed to represent all common music generes but in spite of this, the size of the data set might be the limiting factor. We could overcome this by finding a subset of the original features that preserve the information of 264 features. Since the feature space includes multiple statistics of rhythm and chroma, there could be some correlation that could be exploited to reduce the dimensionality. The correlation heatmap for the training data in figure 2 shows that there is indeed some correlation and especially the seven rhythm statistics are nicely separable.

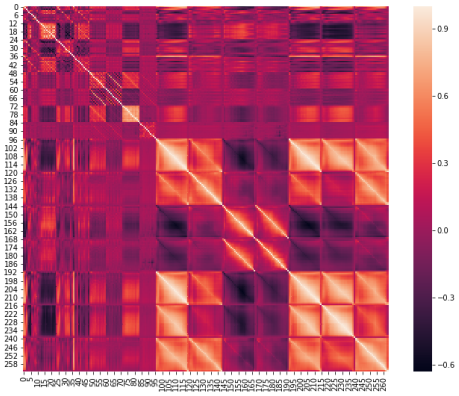


Figure 2. Correlation heatmap

The heatmap is a nice visualization but it would be hard to manually engineer a new feature space that captures the details that matter. For the actual dimensionality reduction task principal component analysis is a much better candidate. The variance explained by PCA is illustrated in figure 3. The figure suggests that the first few principal components explains the majority of the variance and with 10 principal components almost all the variance is explained. The heatmap also indicates that there could be some redundant features so a more interpretable dimensionality reduction technique would be to just discard features that do not have sufficient contribution to the total variance.

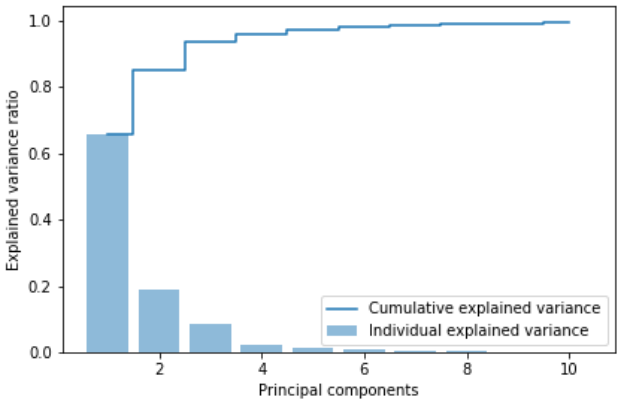


Figure 3. Variance explained by principal components

3. Methods and Experiments

In order to experiment with different ideas and approaches it is important to set up a machine learning pipeline. Such pipeline should take the raw data as input and

produce an accurate description of the quality of the underlying classifier and the classification output when desired. Ideally this pipeline can easily be used to test different classifiers and also to find sufficient hyperparameters for them. Another desirable qualities are speed and accurate estimations of the performance. This section describes how we set up our pipeline and what were the most promising approaches we came up with.

3.1 Pipeline and evaluation methodology

We used pandas framework to read and write the csv-files and relied heavily on scikit-learn to build a robust machine learning pipeline that could effortlessly be used to try different preprocessing steps and machine learning algorithms with sufficient hyperparameters. An example of such pipeline using logistic regression classifier is provided in appendix [ref]

As was mentioned in the previous section, the set of labeled data points is relatively small when considering the classification task. Therefore we decided to use cross-validation instead of a separate validation set. To be more specific, we used stratified k-fold that takes into account the unbalanced class weights. There is no single configuration which we used since we had to do some modifications in order to find the right balance between variance and performance. One of these configurations, for example, was to use 5x3 cross-validation where the data was split into five folds and then the data of the four training folds was once again split into three folds out of which two were used to train the model and one was used to optimize the hyperparameters. This approach resulted in good hyperparameter values and an accurate estimation of the performance.

All in all, our pipeline consists of reading the csv file, splitting the data into appropriate training and validation sets, standardizing the data, possibly reducing the dimensionality and then training and tuning a classifier. The tuned classifier is then either described by some performance metrics or it is used to label the unlabeled data. We relied heavily on this pipeline to evaluate the performance of a classifier instead of submitting the results of every prominent classifier to the Kaggle competition. As a result we might have overlooked some classifier that has been rated against an undesirable or unrepresentative validation set. However, this approach is more likely to work with a real-life machine learning problem.

3.2 Algorithms and methods

Fundamental intuition for choosing models: Multi-class classification being the underlying aspect of the problem statement, we turned to learning models which constitute this feature.

3.3 Logistic Regression

Motivation: Logistic regression though by nature the default for binary classification, the fact that it can be extended for multi-class models was the motivation for choosing it.

Our model predicted the probabilities of different possible genres (labels) corresponding to a given feature space (see the data analysis section). For each of the multi-classification modelling techniques [one-vs-one - *ovo*, one-vs-all - *ova*], we empirically used various algorithms as solvers (like stochastic average gradient) and consecutively varied the following parameters.

- Maximum iterations for convergence $(100-900)$
- Initial class weights (balanced weighing, initializing to 1)
- Weight regularization strength $(0.1-0.9)$
- Tolerance of error across runs

in focus of improving on the accuracy of and reducing loss of the classification.

To optimize the hyper-parameters for modelling the logistic regression classifier we made use of the exhaustive Grid-search cross validation technique.

3.4 Support Vector Machines

Motivation: As the feature-space is of a relatively higher dimensionality, we decided to try out support vector machines to improve on the statistics achieved using the logistic model.

Initially, we modelled a linear support vector classifier which used set of hyperplanes for learning the training data. As we did not observe any drastic improvements in the statistics, we further analysed the data and decided to experiment with the non-linear modelling of the support vector machines.

While empirically using both *ovo* and the *ovr* techniques, we tested various kernels:

- Gaussian radial basis function
- Polynomial (degree 3)
- Sigmoid

For each of the kernels we varied the following parameters:

- Maximum iterations for convergence $(100-300)$
- Initial class weights (balanced weighing, initializing to 1)
- Weight regularization strength $(0.1-0.9)$

- Tolerance of error across runs

To optimize the hyper-parameters for modelling the support vector classifier we made use of the exhaustive Grid-search cross validation technique.

3.5 Ensemble Classifier

Motivation: After experimenting with the above two models for sometime, we figured that logistic regression classifiers were good with accuracy metric and SVMs were minimizing on the log-loss statistic. So we decided to ensemble them together to understand if we can get the better of both worlds.

We used the **voting classifier** strategy to combine the above classifiers using a majority vote to predict the genres. We used both hard and the average predicted probabilities voting (soft vote) methods and found that the hard voting produced better results.

3.6 Semi-Supervised Classifier

Motivation: After not being able to further deduce any meaningful direct relationship between the feature-space, we decided to try the semi-supervised learning methodology.

We chose the label-spreading strategy over label propagation as it is more robust to noise[1]. Modelling steps:

Step 1 Splitting the labelled (train) data as based on some percentage (we chose $\frac{1}{3}^{rd}$ of the it for validation)

Step 2 Merging the split of labelled data from [step-1] with the unlabelled data.

Step 3 Training the classifier/kernel chosen for label-spreading with respective parameters (listed below)

Step 4 Scoring/validating the trained model using the split test data from [step-1]

Step 5 Predicting the genres for unlabelled (actual test) data

We varied the following parameters over runs:

- Kernels (*rbf*, k-nearest neighbours). Varying the gamma parameter (b/w $0.1-00001$) and the number of neighbours attribute (b/w $50-300$) respectively.
- Maximum iterations for convergence $(100-300)$

Other models

Apart from the above mentioned classifiers, we experimented with more models that did not pass our benchmarks for both the accuracy and loss metrics.

- Naive Bayes'
- Decision trees
- Random forests

Though we initially considered reducing the dimensionality by performing principal component analysis, after some substantial scrutiny of the feature space we dropped it. The rhythm bands, pitch classes and the timbre coefficients we spread out across the statistics to discard.

Performance Metrics

Evaluation of all the models designed was done using the k-fold cross validation technique (with $k=5$).

For splitting the data into folds we used stratified process of preserving the percentage of samples for each genre thereby enabling fair evaluation.

4. Results

Performance measures of the above experiments when evaluated with the test data.

4.1 Accuracy, Log-Loss statistics

- Logistic Regression Classifier across various values for its parameters. Results for the best 5 combinations:

Table 1. metrics on test data (LR)

modelling strategy	solver	accuracy	loss
ovr	sag	0.658	0.12
multinomial	sag	0.653	0.17
ovr	liblinear	0.650	0.21
ovr	lbfgs	0.646	0.26
ovr	newton-cg	0.646	0.26

- Support vector classifiers across various values for its parameters. Results for the best 5 combinations:

Table 2. metrics on test data (SVC)

modelling strategy	kernel	accuracy	loss
multinomial	rbf	0.636	0.07
ovr	rbf	0.636	0.07
multinomial	linear	0.607	0.20
ovr	linear	0.607	0.20
ovr	poly (degree 3)	0.567	0.36

- Ensemble classifier with best parametric combination of logistic regression, SVC and voting strategy produces

$$accuracy = 0.631, loss = 0.09$$

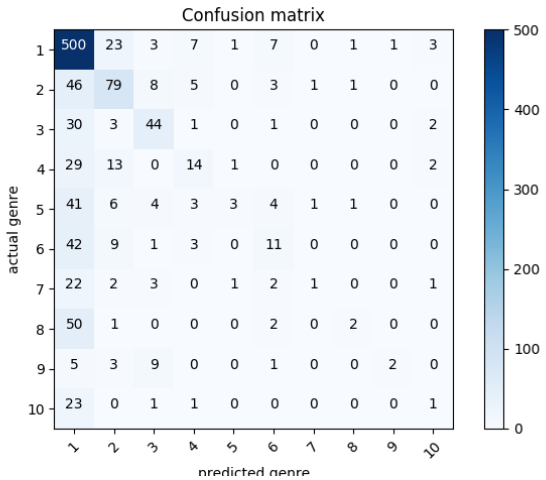
- Semi-supervised classifier with label spreading:

Table 3. metrics on test data (semi-supervised)

kernel	attribute	accuracy	loss
knn	neighbours(300)	0.533	0.14
rbf	$\gamma = 0.001$	0.612	0.29

4.2 Confusion matrix

The following confusion matrix represents the accuracy of classification. It is evident that the prediction is directly influenced by the frequency of respective genres in the training data.



The matrix entry i, j is the number of songs actually in genre i, but predicted to be genre j. Darker the blue, accurate the classification.

5. Discussion

6. Appendices

6.1 Pipeline using scikit-learn

The following code snippet illustrates how easy it is to set up a machine learning pipeline in scikit-learn assuming *load_data* is a function that loads the data set. *LogisticRegressionCV* is a class that trains a logistic regression classifier using all the listed C-values and uses the best-performing classifier to calculate accuracy on the test set. A similar class that can be used to optimize hyperparameters in scikit-learn is *GridSearchCV*.

```
from sklearn.cross_validation import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegressionCV

import load_data

X, y = load_data()

X_train, X_test, y_train, y_test = train_test_spli

classifier = Pipeline([
    ('std', StandardScaler()),
    ('pca', PCA(n_components=100)),
    ('lrcv', LogisticRegressionCV(
        Cs=[0.1, 0.2, 0.5, 0.8, 1, 2, 5]))
])

classifier.fit(X_train, y_train)
accuracy = classifier.score(X_test, y_test)

print(accuracy)
```

References

[1] semi-supervised learning,