Predicting music genre; a review of several machine learning methods

Aalto University
December 9, 2017

***Abstract*—Classification on music genre has been unprecedentedly successful aided with machine learning techniques recently. However, it remains a real challenge to choose a proper method among various learning approaches since genre classification poses three major difficulties, consisting of imbalance, subjectivity and high dimension. This paper reviews mainstream learning methods on this classification task and summarizes their advantages and disadvantages when coping with the difficulties. The methods that have been investigated include linear and non-linear classifiers, dimensionality reduction techniques, and ensemble models. Comparison is drawn based on their accuracy, efficiency, robustness and interpretability. We come to the conclusion that of the tested methods the Support Vector Machine gives the best performance with an accuracy of 65.39%.**

Keywords*—*music genre classification; machine learning

I. Introduction

With the increased use of online media and systems, the importance of Music Information Retrieval (MRI) has grown. One important feature of a song, is the genre. Knowing the genre has a lot of applications, such as advising similar music to users and automatically generating images that match the song.

We all know music has a genre. However, this genre is subjective and this makes it not so easy to classify. It has been shown that even humans don't always agree on the genre [1]. Even while humans do not so bad, manually tagging all songs is a very intensive task looking at the growth of music everyday. Hence, it would be advantageous to develop an automated system for this.

Besides, music genres are unevenly distributed. In the dataset we use, the genre of Pop Rock occupies over half of songs. This imbalance may distort the loss functions of many learning methods, leading to poor predictions on all other genres [2]. Therefore a subtle tuning on the models is needed in order to achieve overall satisfying performance.

What is worse, music genre classification involves high dimensional features, composed of three main components of music: timbre, pitch (melody and harmony) and rhythm. There exists complicated relationship among these features, which makes heavy demands on the classifiers to recognize truly useful indicators.

Therefore, a comprehensive review on proper learning methods for classifying music genres is an urgent need. This review attempts to undertake this task, and is organized as below. First an exploratory analysis is performed on the music genre dataset. Then several mainstream classifiers are probed into, in the order of linear classifiers, non-linear classifiers, dimensionality reduction methods, and ensemble models. At the end the results are shown and discussed.

II. Data analysis

Our training dataset consists of 4363 songs which belong to 10 different genres. Each song is defined by 264 features.

Before selecting models, we have to understand our data since this influences our model selection. We first looked at the distribution of the different classes, shown at Figure 1. As can be seen there are significantly more songs belonging to the first class than to the other classes. We have to take this into consideration for our models. Because of this uneven distribution, there is a chance that our model mainly predicts class 1 (Pop Rock) fed with all the data, since this has in general the highest probability of being right.

We also looked at the density distributions for all the features. From here it can be seen that many features have a Gaussian shape. Moreover, some have a lot of occurrences of one value. Some density functions of features are really similar, i.e. have a high correlation, which indicate that some features could maybe be left out to get the same or possibly even better results with the models.

We also looked at the correlations between the labels and all the features. It turns out most

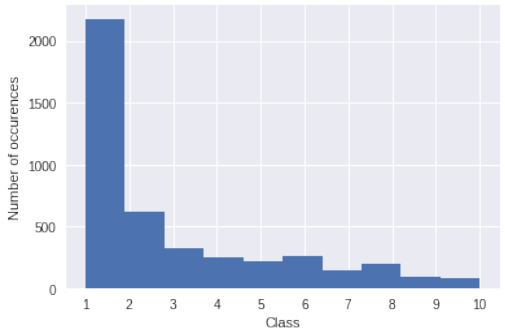

Fig. 1. Distribution of the classes in the training dataset

correlations are rather small. On average the absolute correlation is 0.125 and the highest absolute correlation is 0.382. However, when making a scatter plot of the labels and the features with a high correlation, there is still no clear relation visible. See for an example, figure 2 which depicts the scatter plot of the classes and the feature with the highest correlation to the classes. The insignificantly high correlations indicate that we will probably be unable to make the prediction with only a small set of features, at least linearly. This analysis is useful for the dimensionality reduction, elaborated at the methods and experiments section.

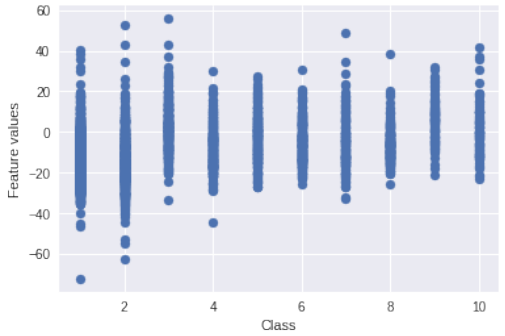

Fig. 2. Scatter plot of the classes and the feature with the highest correlation

III. Methods and Experiments

A. Linear Classifiers

The simplest classifier is a linear classifier, based on which many advanced methods are constructed. They are known for their simplicity, efficiency, and robustness. However, they are only capable of capturing linear transformation between the input and the output, and unfortunately, relationships among real life entries are mostly non-linear.

The most well known linear classifier is the logistic classifier [3]. It can be treated as one linear transformation plus one sigmoid function to produce probabilities for discrete classification. To be more precise, in logistic regression we try to predict the class y by the predictor h where $\hat{y} = 1$ if $h > 0.5$ and $\hat{y} = 0$ else.

$$h_i = \sigma(\mathbf{x}_i^T \mathbf{w}) = 1/(1 + \exp(-\mathbf{x}_i^T \mathbf{w})).$$

The goal is to learn the optimal weight matrix $\mathbf{w}$ which minimizes the logistic loss, which is defined as

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} \ln\left(1+\exp\left(-y_i(\mathbf{x}_i^T \mathbf{w})\right)\right).$$

Since our problem entails multiple classes, we train 10 classifiers. Each of them gives a probability of the sample belonging to that class or any to the other classes, and class with the highest probability is chosen as the predicted label for that sample, which is also known as the one-vs-all technique.

In order to combat overfitting, we add a L2-regularization that forces the weights to not get too big. Including the L2-regularization, our full optimization problem becomes

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} \ln\left(1 + \exp\left(-y_i(\mathbf{x}_i^T \mathbf{w})\right)\right) + \lambda ||\mathbf{w}||^2.$$

We choose logistic regression as one of our models, since it is such a simple model but has been proven to work rather well in many applications.

B. Generalized Additive Models (GAMs)

A linear classifier can be adjusted to move beyond linearity, while still possessing the advantages of the linear world. Generalized Additive Models (GAMs) [4] provide a general framework for extending a standard linear model by allowing non-linear functions of each of the variables, while maintaining additivity.

$$\hat{y}_i = \sigma(\beta_0 + \sum_{i=1}^{N} f_i(x_i))$$

where $f_i(x_i)$ can be arbitrary non-linear functions. For this specific task, a logistic classifier extended

with two degree polynomial transformation on features is employed.

C. Non-linear Classifiers

Non-linear classifiers are blessed with more powerful capability than linear ones, since they are able to take into account non-linear relationships among indicators. However, their advantages are established at the expense of overfitting and hyperparameters.
The most famous non-linear classifier are Support Vector Machines [5], which maximizes a margin between two different groups to ensure a lower variance by minimizing the following loss function

$$\frac{1}{N}\sum_{i=1}^{N} max(0, 1 - y_i(\mathbf{w}\mathbf{x}_i - b)) + \lambda||\mathbf{w}||^2$$

Besides, it makes use of various kernel functions to obtain not only non-linear ability but also an efficient optimization solution [6]. Since we expect high non-linearity in the data, we implement a SVM with a radial basis function (RBF) kernel. A disadvantage of SVM with a RBF kernel is that it has two hyperparameters and it is not trivial to find optimal values for them. Hence, we performed a grid search with K-fold cross validation. First hyperparameters are searched in a logarithmic scale, and then we successful narrowed the range down to [1,10] for C and [0.001, 1] for gamma.
Last but not least, for the decision function, we use one-vs-rest.

D. Ensemble Models

A single model is mostly performed for academic purposes, while in real applications, an ensemble model [7] is preferred, since it averages the results from multiple different learning methods to improve accuracy in prediction. It can be proved that with k uncorrelated models, the expected square of error ϵ can be reduced to $\frac{1}{k}\epsilon^2$ [8]. In theory, any learning method can benefit from ensemble models at the price of increased computation and memory.
In this case, XGBoost [9] is selected for the task. XGBoost applies a boosting technique [7] to construct an ensemble with higher capacity than the individual models. Particularly, it uses a decision tree as its basic elemental model, and keeps building trees one by one, using an additive strategy, that is, to fix what have been learned. After the tree ensembles, it is usually followed by a process of pruning to avoid overfitting. Although XGBoost is extremely powerful, it requires a rigorous tuning among over tens of hyperparameters, which is difficult even for a small dataset in this task.

E. Data normalization

It depends on the data and model, but in general it is good practice to normalize the data [3]. In order to see if normalizing the data is beneficial in our case, we compare the performance of logistic regression with and without normalized data. Based on the effect we decide whether to also normalize the data for our other models. By normalization it means that we standardize the data to have a zero mean and unit variance.

F. Dimensionality Reduction

Dimensionality Reduction (DR) is of great interest since real life data such as social network and images often suffer from "the curse of high dimensionality" [10]. DR is mostly applied for purposes such as lower dimensional embedding, visualization and features selection. Moreover, DR techniques such as Locally linear Embedding, Isomap, and t-SNE are able to accomplish manifold learning, unfolding a high dimensional object into lower dimension while preserving its structure [11].
For the task in the review, we adopted the most famous DR method, Principal Component Analysis (PCA) [12], a linear dimensionality reduction technique that minimizes the reconstruction error when reducing the dimension.

$$E_{codec} = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{y_i} - \mathbf{dec}(\mathbf{cod}(\mathbf{y_i}))||^{\mathbf{2}}$$

A line search is employed in order to find the hyperparameter k for PCA, the number of components after reduction. The reduced data is then used to train the model that so far reached the highest accuracy on the test set in order to see if the performance increases when using PCA.

G. Under- and oversampling

An important characteristic off our dataset that can influence our models is the uneven distribution of samples per class. To combat this, we can under- and/or oversample our data and train the model on this new data. When undersampling, we take the class with the lowest number of samples and reduce the number of samples in the other

classes to this number. When oversampling, we take the class with the highest number of samples and duplicate our samples in the other classes, to get in all classes the same number of samples. We apply those two methods plus a more sophisticated method called SMOTETomek. This is a combination of an oversampling method named Synthetic Minority Over-Sampling Technique (SMOTE) and an undersampling method called Tomek links. SMOTE generates new samples based on nearest neighbours and a random vector [13], hence it creates new samples instead of duplicating existing samples. A disadvantage of this method, is that the created sample space can be quite messy and hence we use Tomek links to clean up the space a bit. The basic idea between Tomek links is to remove the samples from the majority classes which are close to the minority classes. This generates a more separated cluster space which often increases the classification performance. We use a ratio of 0.8 which means that we look at the class with the most samples, in this case class 1 with 1955 samples, and increase the number of samples in the other classes to 0.8 of that number, in this case 1564. Moreover, we set the number of nearest neighbours that are used in SMOTE to 5. Again, we will apply these methods on the model that gives the highest accuracy on the test set.

H. Model evaluation

We evaluate our models firstly by the accuracy, defined as

$$accuracy = \frac{y_{true} == y_{predicted}}{N}$$

where N is the number of samples, y_{true} the real class and $y_{predicted}$ the predicted class. We use accuracy since it is a very common and simple evaluation metric. We assess the accuracy on the training set by 5-fold cross-validation on the whole training set. The accuracy on the test set is computed via Kaggle. When we refer to the best model, we in principle refer to the one with the highest accuracy.
However, we also evaluate our models by looking at the multiclass log-loss on the test set. This is defined as

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j}).$$

Since we are predicting several classes, we also look at the confusion matrix, in order to get a better image of which classes the model can predict well and which ones it is having trouble with.

Classifier	Train set accuracy	Test set accuracy	Test set log-loss
Logistic Regression, not normalized	0.64541	0.63768	1.49866
Logistic Regression, normalized	0.64886	0.64394	1.48107
Logistic classifier, 2 degree polynomial	0.63917	0.61888	1.50106
Support Vector Machine, C=2.9, gamma=$\frac{1}{264}$	0.66283	0.65388	0.16907
XGBoost	0.67698	0.65158	0.17507
SVM, PCA=150	0.64496	0.64364	0.17830
SVM, SMOTE-Tomek	0.66582	0.63646	0.17649

TABLE I. ACCURIES FOR THE DIFFERENT MODELS ON THE TRAINING AND TEST SET

IV. RESULTS

See Table I for the accuracies of the different methods on the training and test set and the log-loss on the test set. The accuracies on the training set are from 5-fold cross-validation on the whole training set. The accuracies and log-loss on the test set are the results of the submissions to Kaggle.

A. Normalizing data

We implemented logistic regression for unnormalized and normalized data. As can be seen, the accuracy on normalized data is higher and hence for all other models normalized data was used by default.

B. Hyperparameters of SVM

After grid search, the best values for the hyperparameters we found were C=2.9 and gamma=$\frac{1}{\text{number of features}} = \frac{1}{264}$.

C. Number of components for PCA

In order to decide upon the number of components to use for the dimensionality reduction with PCA, we looked at the explained variance of the components. See Fig 3 for the explained variance per component and the cumulative sum. After a line search, top 150 components are chosen, which can be seen explaining 0.98448% of the

variance of the original data. Since SVM is our best performing model, we applied PCA to this model.

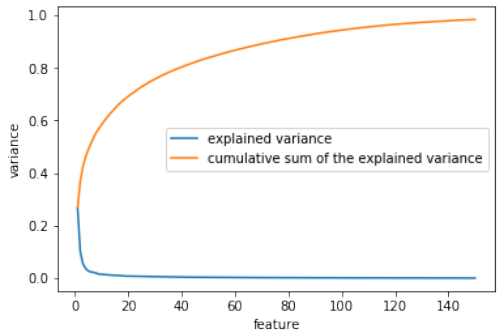

Fig. 3. Plot of the explained variance versus the components of PCA.

D. Over- and undersampling methods

As described, we tested 3 methods. However, the naive over- and undersampling method performed significantly worse, so they are not reported here. We applied all 3 methods on the SVM model, since this was our best performing model.

E. Confusion matrices

See Fig. 4 and Fig. 5 for the normalized confusion matrices of the SVM and SVM with SMOTETomek respectively.

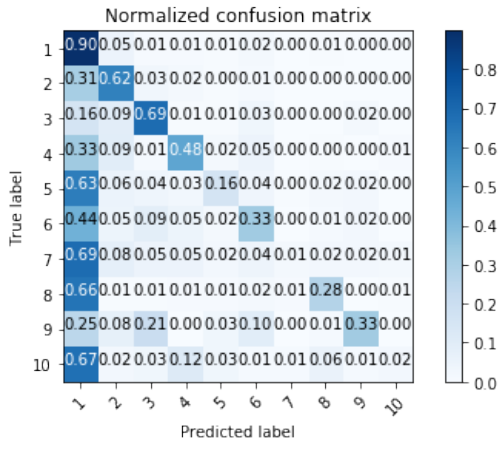

Fig. 4. Normalized confusion matrix of the SVM.

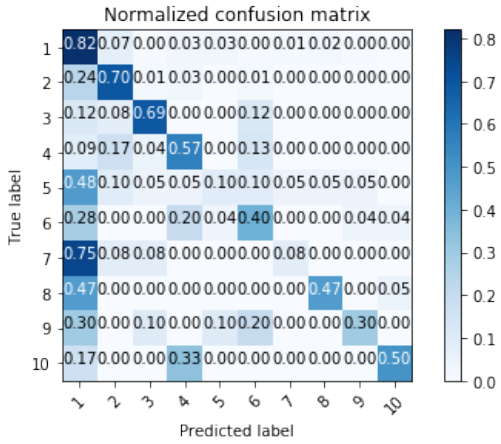

Fig. 5. Normalized confusion matrix of the SVM with as input data sampled by the SMOTETomek method.

V. Conslusion and Discussion

As can be seen from the results, SVM gives the best performance of the models that we have tested.

We were quite surprised that a simple logistic regression model performs so well, giving only 2% less accuracy than our best performing model. Apparently it is rather easy to separate the classes to a certain extent, after which even a small improvement has become hard. For instance, when we enhanced it with twp degree polynomial GAM, however it only achieved an accuracy of 61.888%, while took more than 2 hours to run on a personal laptop, which means the high dimension and complex relationship among features hinder a further improvement on this task. Also the promising method of XGBoost doesn't perform better than SVM, though for XGBoost an increase in performances might be possible by tuning the hyperparameters more.

When decreasing the dimensionality from 264 to 150 with PCA, the accuracy decreases. Apparently a linear reduction technique is incapable of describing the relationship among music genres features, even though our 150 features have explained 0.98% of the variance.

Also the over- and undersampling method did not improve the performance on the test set. It did improve the performance a bit on the training set, but apparently this was not beneficial for unseen data. An interesting observation that we can see from the confusion matrices is that using SMOTETomek does improve the correctly clas-

sified samples from several classes with a small number of samples. However, it does decrease, for example, the performance on prediction of class 1 quite a bit.

There were no clear differences in performance between accuracy and log-loss, except for the logistic regression models. The logistic regression models have an exceptionally high log-loss, while normally these models have a low log-loss compared to other models. We tried to understand the reason behind our high log-loss, but we couldn't find a logical explanation.

Due to our imbalanced dataset, accuracy and log-loss might have not been the best performance metrics. For example, recall and precision are scores that often give a better image of the real performance of imbalanced datasets.

For future research, more models could be compared, such as neural networks and ensembles of methods. Moreover, more time should be put into finding the optimal hyperparameters for SVM and XGBoost.

REFERENCES

[1] S. Lippens, J.-P. Martens, and T. De Mulder, "A comparison of human and automatic musical genre classification," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 4. IEEE, 2004, pp. iv–iv.

[2] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser, "Svms modeling for highly imbalanced classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 281–288, 2009.

[3] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1.

[4] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[5] C. Cortes and V. Vapnik, "Support vector machine," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[6] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.

[7] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[9] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[10] R. Bellman, *Dynamic programming*. Courier Corporation, 2013.

[11] J. A. Lee and M. Verleysen, *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[12] I. T. Jolliffe, "Principal component analysis and factor analysis," in *Principal component analysis*. Springer, 1986, pp. 115–128.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

VI. APPENDIX

See below the code for our most important basic models.

A. Normalization

```
import numpy as np
from sklearn.preprocessing import
    StandardScaler

xtrain = np.loadtxt('Data/train_data.
    csv', delimiter=',')
ytrain = np.loadtxt('Data/
    train_labels.csv', delimiter=',')
xtest = np.loadtxt('Data/test_data.
    csv', delimiter=',')

scaler = StandardScaler()
xtrain_nl = scaler.fit_transform(
    xtrain)
xtest_nl = scaler.fit_transform(xtest
    )
```

B. Logistic Regression

```
from sklearn.linear_model import
    LogisticRegression

model_nl = LogisticRegression()
model_nl = model_nl.fit(xtrain_nl,
    ytrain)
ypred_nl = model_nl.predict(xtest_nl)
```

C. SVM

```
from sklearn import svm

C_,gamma_ = 2.9,0.0046415888336127772
clf = svm.SVC(decision_function_shape
    ='ovr', random_state=42, C=C_,
    gamma=gamma_)
fit = clf.fit(xtrain, ytrain)
ypred = fit.predict(xtest)
```

D. XGBoost

```
import xgboost as xgb

parameters = {
    'learning_rate': [0.1], #so
        called `eta` value
```

```
    'max_depth': [6],
    'min_child_weight': [1.5],
    'gamma': [0.5,1.0],
    'silent': [1],
    'subsample': [0.9],
    'colsample_bytree': [1],
    'n_estimators': [300], #number of
        trees
    'seed': [42]}
xgb_model = xgb.XGBClassifier()
clf = GridSearchCV(xgb_model,
    parameters,
                   cv=3,
                   scoring='accuracy'
                       ,
                   verbose=1, refit=
                       True)
clf.fit(xtrain_s, ytrain_s)


yprob = clf.best_estimator_.
    predict_proba(xtest)
```

E. PCA

```
from sklearn.decomposition import PCA
from sklearn import svm

pca = PCA(n_components=150)
pca.fit(xtrain)
xtrain_pca = pca.transform(xtrain)
xtest_pca = pca.transform(xtest)

C_ = 2.9
clf = svm.SVC(decision_function_shape
    ='ovo', random_state=42, C=C_)
fit = clf.fit(xtrain_pca, ytrain)
ypred = fit.predict(xtest_pca)
```