Analysis of Music Genre Classification Based on Machine Learning

Anonymous

1 Introduction

At present, traditional manual retrieval methods can no longer satisfy the retrieval and classification of the massive information of music genres, therefore, computer automatic music type classification has become an important part of multimedia applications. With the continuous development of machine learning technology, various classification technologies based on machine learning have achieved different levels of success. Music classification is essentially a pattern recognition problem, which mainly includes two aspects: feature extraction and classification. This article mainly focus on the processing of existing feature data and the subsequent classification process. The genres involved include: Soul and Reggae, Pop, Punk, Jazz and Blues, Dance and Electronica, Folk, Classic Pop and Rock and Metal.

2 Data analysis and preprocessing

Data analysis and preprocessing are very first and foremost steps before classification. Some basic facts of the problem can be intuitively read from the training data set: as shown by the training data, this music genre classification is a multi-classification problem with quantitative features. By observing the inherent defects of data sets, some preprocessing methods can be proposed, and these methods can always greatly enhance the performance of the final classifier. At the same time, feature analysis is also indispensable. We classify and extract features to make them suitable for different classifiers, so as to achieve the best classification performance.

2.1 Data analysis

The training set is composed of 7668 instances (Bertin-Mahieux et al., 2011) which is a satisfactory large scale data set. Sufficient training data means that over fitting due to insufficient training is unlikely to occur. However, labels of this training set is obvious imbalance, in other words, instances with some labels such as folk are superfluous and instances with some other labels (jazz and blues, etc.) are too few (see Figure 1). Imbalanced training set leads to the model's insufficient training for some specific labels, so that the classification performance on these label is extraordinarily poor. Therefore, training samples need to be assigned with appropriate weight to alleviate the imbalance problem in the subsequent classification process.

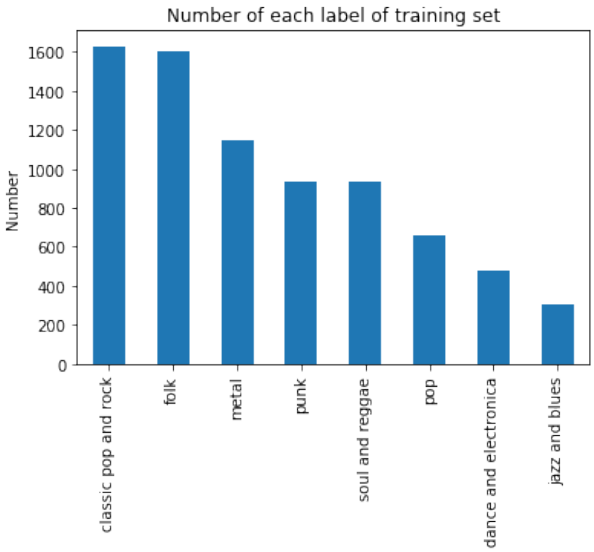

Figure 1: Imbalanced training set

2.2 Feature analysis

The existing features is mainly divided into two categories: textual and numerical features. Obviously, the track ID in the numerical feature contains no meaningful information for classification, because each instance has its unique ID. Therefore, we will discard this feature in subsequent studies. In textual features, compared with lyrics, song titles are almost specific feature, which is not containing helpful information for subsequent generalization as well. On

the contrary, apparently, lyrics are of high value for classification. According to common sense, the words appearing in lyrics can often represent the style of the music, and thus are related to the category of the music.

In numerical features, loudness, rhythm, tune, duration, size, and specific audio characteristics(Schindler and Rauber., 2012) are all related to music genre, which means all of these features need to be considered in the following study.

2.3 Data preprocessing

For text features like lyric tags, one-hot transformation is a common treatment methods. For continuous numeric data, different value ranges will cause the gradient descent method and the KNN method to have inconsistent weights on different features which may hugely affects the performance of the model. Therefore, it is necessary to perform a scaling/normalization on this type of features before training the model.

3 Classification models and their Evaluation

Scikit-learn provides a batch of classifiers that can be easily implemented(Pedregosa et al., 2011). For different types of problems and feature data, different classifiers need to be implemented to achieve the best prediction accuracy(see Figure 2)(Buitinck et al., 2013). In the following research, several representative classifiers will be analyzed and evaluated.

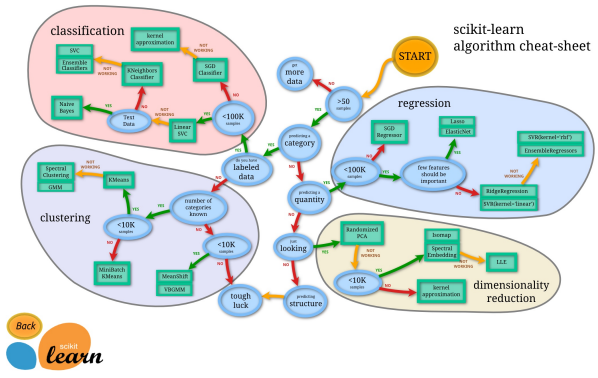

Figure 2: Scikit-learn official guide: how to choose classifier

It is worth mentioning that 0-R classifier is using as the baseline classifier. Accuracy of 0-R on training set and valid set are respectively 0.21 and 0.12 with most frequent strategy.

3.1 Naive Bayes Classifier

Naive Bayes classifier(Rish and others, 2001) for multinomial models (MNB classifier) is suitable for classification with discrete features (e.g., word counts for text classification).[1] Naive Bayes assumes that lyric features are independent of each other, although they may be related. As a result of our experiments, multinomial Naive Bayes classifier performances strongly when training lyric tags data with Laplace smoothing (see Figure 3). As shown in

	Soul..	Pop	Punk	Jazz..	Dance..	Folk	Classic..	Metal
Soul..	41	0	0	5	2	3	6	1
Pop	0	74	0	0	0	0	0	0
Punk	1	0	31	1	4	1	2	4
Jazz..	2	3	1	10	1	5	18	4
Dance..	9	0	1	6	6	12	9	2
Folk	1	0	0	5	0	39	13	6
Classic..	14	0	0	16	4	6	15	0
Metal	5	2	2	0	8	2	5	42

Figure 3: Confusion matrix of Naive Bayes with bagging

the figure, when number of training instances increasing, training accuracy and validation accuracy converge to around 0.60 (see Figure 4). Obviously, despite the large number of features, this model does not show significantly overfitting.

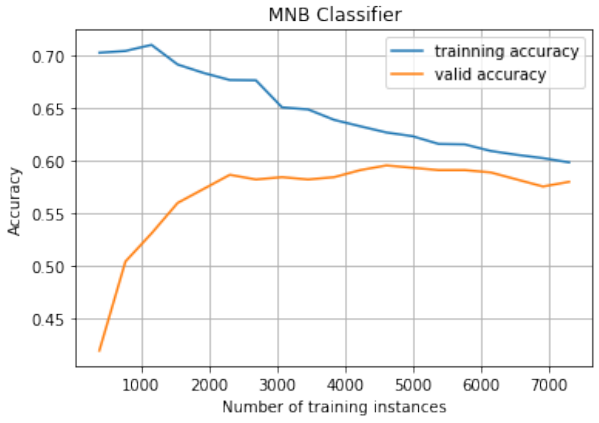

Figure 4: MNB classifier performance vs. different scale of training instances

3.2 K-Nearest Neighbours Classifier

K-Nearest Neighbours Classifier (KNN classifier) finds K sample data closest to the test data,

[1]Scikit learn official guide

and judges the type of test data on merging of categories the K closest samples (Cunningham and Delany, 2020). When using KNN classifier, the data should be normalized to make influence of each feature on the result tends to be balanced. To be precise, KNN does not have a real training model, which leads to its fast training speed but relatively slow prediction speed.

For KNN, the value of K is a particularly critical factor. A smaller value of K will reduce the approximation error of training, but the estimation error of prediction will increase, which is prone to over-fitting. A larger value of k will reduce the estimation error of prediction, but the approximation error of training will increase. When K is too large, the model becomes simple with large bias and low variance in an under-fitting state. After verification, K=11 is a appropriate value with satisfied approximation and estimation error at the same time (see Figure 5). This value is consistent with the conclusion we got using the gridSearch method from sklearn library.

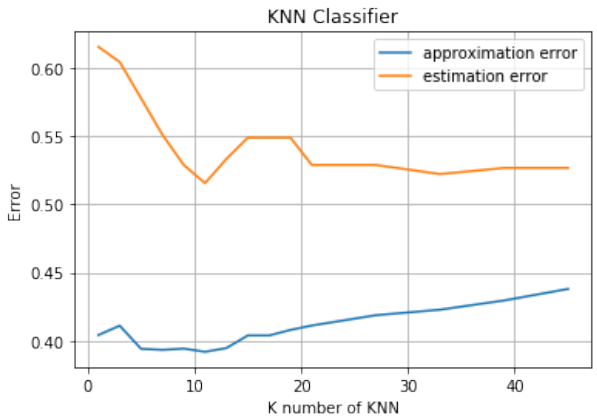

Figure 5: K value vs. Error

3.3 Multi-layer Perceptron Classifier

A multilayer perceptron (MLP) is a class of artificial neural network(Wikipedia contributors, nd). For MLP classifier, the most critical parameter is the structure of the perceptron network, namely number of hidden layers (depth of neural network) and parallel neurons in the hidden layer (width of neural network), which will directly affect the performance of the MLP classifier. unfortunately, generally the way to judge the number of nodes is tricky and ambiguous. For this model, we did some analytical experiments to choose the factors. According to the training error and test error of different node numbers, it can be seen that when there are too few nodes, the model is under-fitting; when there are too many nodes, the model has a trend of over-fitting (see Figure 5). Therefore, we choose the appropriate number of nodes around 256 in the middle. Limited by the computational power, we finally choose two-layer neural network with 256 nodes each layer.

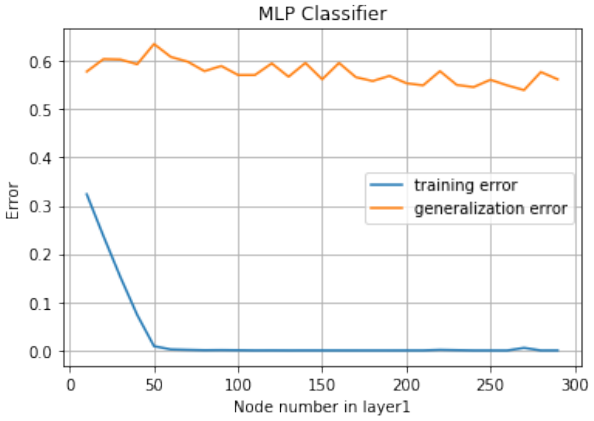

Figure 6: Number of Nodes in First Layer vs. Error

Another critical parameter is activation function for the hidden layer. According to GridSearch in scikit learn, 'relu', the rectified linear unit function is the best choice among all the activation functions.

However, even if the appropriate parameters are selected, the MLP classifier still has an obvious tendency to overfit. The Bagging method can alleviate this tendency of overfitting, and thus get better prediction results (see Table 1).

	training acc.	valid acc.
MLP with bagging	0.98	0.50
MLP without bagging	1.0	0.45

Table 1: Comparison with and without bagging

3.4 Decision Tree

Decision tree is to build a classification model with a tree structure. Each node represents an feature. According to the division of this feature, it enters the child nodes of this node until reaching the leaf nodes. Each leaf node represents a certain category/label, so as to achieve Purpose of classification.

3.4.1 Decision Tree with bagging

According to the characteristics of decision tree, decision tree will almost inevitably over fit. We can alleviate this problem by limiting the number of nodes or setting a threshold to limit tree growth. For an over-fitting decision tree, pruning can be performed, such as pruning nodes that may increase the validation error. Here, we use a more general method to alleviate over fitting: bagging. According to the data, the bagging method has a considerable positive effect on the decision tree (see Table 2).

	training acc.	valid acc.
DT with bagging	1.0	0.63
DT without bagging	1.0	0.41

Table 2: Comparison with and without bagging

	Soul..	Pop	Punk	Jazz..	Dance..	Folk	Classic..	Metal
Soul..	42	0	0	0	0	10	6	0
Pop	0	70	4	0	0	0	0	0
Punk	0	2	33	0	0	5	4	0
Jazz..	3	0	1	10	1	19	9	1
Dance..	8	1	3	0	10	8	10	5
Folk	1	0	1	1	0	38	22	1
Classic..	1	0	4	2	0	21	27	0
Metal	0	0	11	0	0	1	0	54

Figure 7: Confusion Matrix of Decision Tree with bagging

3.4.2 Random Forest Classifier

Random forest is actually a special bagging method that uses decision trees as a sub model in bagging (Pal, 2005). It integrates the idea of ensemble learning, bootstrap and bagging. When building decision treerandom forest randomly samples both instances and features. Compared with common decision tree, RF improves the accuracy of valid set and reduces the over-fitting. However, the confusion matrix shown that although weight is introduced in training to solve the imbalanced, the recall value of a specific label, such as jazz, is extremely low (see Figure 8), which may significantly affect the final prediction accuracy. Therefore, we avoid to select this classifier in following stacking of multiple models.

	Soul..	Pop	Punk	Jazz..	Dance..	Folk	Classic..	Metal
Soul..	40	0	0	0	1	11	6	0
Pop	0	73	0	0	0	1	0	0
Punk	0	1	29	0	0	4	4	6
Jazz..	1	0	1	0	0	32	9	1
Dance..	6	1	5	0	7	9	12	5
Folk	0	0	1	0	0	39	24	0
Classic..	2	0	0	0	0	23	29	1
Metal	0	0	0	0	0	0	1	65

Figure 8: Confusion Matrix of Random Forest

3.5 Boosting

The Boosting algorithm optimizes the classification results through a series of iterations, each iteration introduces a weak classifier to overcome the existing shortcomings of weak classifier combinations (Schapire, 2003). Some common seen boosting classifiers are Adaboost and Gradient Boosting Classifier.

Compared with bagging, the main difference between this two methods is the sampling method. Training set selection of bagging is random, in contrast, selection of the training set of each round of boosting is related to the learning results of the previous rounds. Each prediction function of bagging has no weight, while boost has weight. In bagging, each function can be generated in parallel, while in boosting prediction functions can only be generated sequentially. Therefore, the performance of boosting is often surpasses bagging, while concomitantly, boosting is more likely to be time-consuming.

3.5.1 Gradient Boosting Classifier

Gradient Boosting is a method of Boosting. Its main intuition is that the model built later is at the gradient descent direction of the loss function of the model built before. The Gradient Boosting classifier in scikit-learn uses an ensemble of weak prediction models such as decision trees. In the case of using the same training data, its performance surpasses Random Forest using the bagging method in overall predict accuracy as well as uniformity of precision/recall on each label (see Figure 9), nevertheless, it takes much longer (see Table 3).

	valid acc.	traning time
Random Forest	0.64	24s
Gradient Boosting	1.0	1476s

Table 3: Comparison boosting and bagging

	Soul..	Pop	Punk	Jazz..	Dance..	Folk	Classic..	Metal
Soul..	42	0	0	0	9	0	7	0
Pop	0	72	1	1	0	0	0	0
Punk	0	1	35	2	0	4	2	0
Jazz..	2	3	2	17	2	12	5	1
Dance..	2	0	6	3	19	7	4	4
Folk	1	0	0	0	1	45	17	0
Classic..	9	0	4	2	3	12	25	0
Metal	0	0	14	0	0	0	1	51

Figure 9: Confusion Matrix of Gradient Boosting Classifier

3.6 Support Vector Machines Classifier

In principle, support-vector machine constructs a hyperplane or set of hyperplanes in a high or infinite-dimensional space, which can be used for classification (Hearst et al., 1998). In this task, we are using SVC (Support Vector Classifier) with non-linear classification in scikit-learn. The overall accuracy of validation is 0.66, what's more, performance of each category is relatively balanced and there is no particularly bad performance.

	Soul..	Pop	Punk	Jazz..	Dance..	Folk	Classic..	Metal
Soul..	37	0	0	3	15	1	2	0
Pop	0	73	0	0	0	0	1	0
Punk	0	3	33	0	0	3	4	1
Jazz..	3	0	2	16	3	13	6	1
Dance..	5	3	7	2	15	6	4	3
Folk	1	0	3	1	0	37	22	0
Classic..	2	0	1	13	0	7	32	0
Metal	0	1	7	0	2	0	0	56

Figure 10: Confusion Matrix of Support Vector Machines Classifier

4 Stacking of multiple classifier

Stacking is a common method of ensemble learning. The intuition of stacking is smooth errors over a range of classifiers with different biases. Since we have evaluated several different classifier, for reaching a higher overall performance, it is natural to think of trying to use the stacking method to merge multiple models. Although stacking can not bring much research value, it is often a quite effective method in engineering practice. When choosing a classifier for stacking, there are usually the following points to consider: the accuracy of the model itself; the independence between model errors; and the number of models. When the errors of classifiers are independent of each other, stacking can get the best effect. At the same time, major voting should use an odd number of classifiers.

	NB	KNN	MLP	DT	RF	GB	SVM
NB	1	0.25	0.28	0.35	0.38	0.46	0.38
KNN	0.25	1	0.58	0.34	0.30	0.28	0.35
MLP	0.28	0.58	1	0.29	0.22	0.40	0.35
DT	0.35	0.34	0.29	1	0.58	0.47	0.50
RF	0.38	0.30	0.22	0.58	1	0.45	0.52
GB	0.46	0.28	0.40	0.47	0.45	1	0.54
SVM	0.35	0.35	0.35	0.50	0.52	0.54	1

Table 4: Independence of classifiers errors (0-independent;1-identical)

Stacking here selects three classifiers with the highest accuracy and respectively low dependence, including Naive Bayes, Decision Tree with bagging and Support Vector Machine (see Table 5). When there is a tie, which means every single classifier gives different result of prediction, we choose to use prediction of highest accuracy, namely SVC. Through observation, performance of meta classification is not as good as that of major voting, so voting is finally used to determine the prediction result. Finally, the stacking classifier's accuracy over valid set is 0.7, which is higher than all of the individual sub classifiers before stacking.

	training features	valid accuracy
0-R Baseline	-	0.122
Naive Bayes	lyric	0.573
KNN	numeric	0.484
MLP	numeric	0.493
Decision Tree	all	0.631
Random Forest	all	0.627
Gradient Boosting	all	0.680
SVM	all	0.664
Stacking	all	0.700

Table 5: Comparison of all Classifiers

5 Conclusions

For this music genre the classification problem, we first analyze and preprocesse the data. Subsequently, some typical classifiers was analyzed and evaluated using given music genre data set. After that, we select the best model after comparing all evaluated classifiers and integrate them into one classifier using ensemble method, and finally reached an accuracy of 0.648 in the kaggle competition. In general, our classifier based on machine learning technology has a satisfactory performance when predicting

unknown genres of music. At the same time, we have reason to believe that for this music genre classification issue, such performance is far from the upper limit.The classifiers using different methods still have much room for expectant improvement and development.

References

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. 2011. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Padraig Cunningham and Sarah Jane Delany. 2020. k-nearest neighbour classifiers–. *arXiv preprint arXiv:2004.04523*.

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

Mahesh Pal. 2005. Random forest classifier for remote sensing classification. *International journal of remote sensing*, 26(1):217–222.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Irina Rish et al. 2001. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46.

Robert E Schapire. 2003. The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer.

A. Schindler and A. Rauber. 2012. Capturing the temporal domain in echonest features for improved classification effectiveness. In *Proceedings of the 10th InternationalWorkshop on Adaptive Multimedia Retrieval (AMR 2012)*.

Wikipedia contributors. n.d. Wikipedia:Lists of common misspellings. In *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Wikipedia:Lists_of_common_misspellings&oldid=813410985.