Music genre classification
COMP90049 Report

Anonymous

1 Introduction

The task of music classification has numerous applications ranging from music recommendation to music discovery, copyrights infringement detection to next-hit prediction.

Using the Million Song Dataset Challenge (Bertin-Mahieux et al. 2011), specifically the features extracted from these songs with Echo-nest (A. Schindler and A. Rauber 2012), this paper aims to develop a music genre classifier model, using some standard techniques from Machine learning.

Furthermore, how some feature engineering and feature selection techniques could impact accuracy of different models is studied.

2 Related work

The Million Song Dataset is used by several researchers to develop and test different models to auto-tagging, song recommendation and genre classification.

Much research has been done into feature engineer and selection for music genre classification tasks. Several works have been developed in the topic of feature extraction from audio signal, trying to reduce the size of the sample audio required to identify a genre for a particular song. For instance, some research have shown the use of *Wavelet coefficient histograms* can increase predictability over combination of features such as *Mel-Frequency Cespatral Coefficients* (MFCC) and Pitch (Li, Ogihara and Li, 2003), but these features are captured from the audio signal, which is not possible in many circumstances.

2.1 Feature selection

In their original paper, Schindler and Rauber (2012) used the feature extracted Echo-Nest to demonstrate that those features were similar to MFCC and Pitch metrics, that are commonly used for classification. They also implement a *Temporal Domain* feature to enhance predictability of audio features by using statistical metrics including mean, median, variance, min, max, value range, skewness, kurtosis of loudness, timbre, pitches, and other metrics delivered by Echo-nest API. With those calculation, they were able to train music genre classification models with variations of *Support Vector Machines* and *K-Nearest Neighbor* algorithms, achieving accuracies ranging from 50% to 89% depending on the dataset source.

Other authors have tried to create *Temporal Features Integration* by applying autoregression over short time (such as MFCC), medium time and long time features and then use them as new features to classify song into genre using lineal or Gaussian classifiers (Meng, Ahrendt, Larsen and Hansen, 2007).

However, for the present case the name of the features are unknown, the summary metrics are hard to calculate as it is difficult to determine when a group of vector belong to a particular metric using other information than scales of each feature and correlation values. Therefore, other strategies are required to select or summarized features.

2.2 Algorithms used in literature

As mentioned before, *Support Vector Machine* are used in several papers, which is an algorithm tries to find a line or plane that separate the different classes. To do so, the algorithm uses kernels to transform the data in such a fashion that such line can be draw. For better generalization, the algorithm tries to maximize the distance to the closest element of each class, drawing support vectors (Cortes and Vapnik, 1995).

Other algorithms present in literature that seems suitable to our task are *Decision Trees, K-Nearest Neighbor, Logistic Regression* and *Randoms Forest*. (Tan, Steinbach, Karpatne and Kumar, 2005).

3 Dataset description and exploration

The dataset consists in textual, numerical, and categorical attributes. This is a high-dimensional dataset, which has implications for some classification algorithms traditionally use in the field of machine

learning. Algorithms such as *Decision Tree* could overfit if all features are considered but it could lose important information only few are counted. High dimensionality may have impact in high time-complexity algorithms such as *K-Nearest Neighbor (k-NN)*.

The textual attributes consist in “song title” and “tags” which represent unique word from lyrics. Some songs have a few tags while other have over 300.

Numerical attributes can be divided into known features and unknown vectors. Unknown vectors are 148 attributes in a variety of scales. However, several of these features have some high correlation between them. This may have some implications over *Naïve Bayes classifier* as many features seems to be dependent, then some conditional probabilities could be overestimated. Also, scale diversity has implication on *k-NN* algorithms, as similarity calculations could be affected by features with higher magnitude.

Finally, the dataset classes are eight genres, therefore this is a multiclass task where classes are imbalanced. For instance, genres *Dance and Electronica* and *Jazz And Blues* have less than 500 instances, while classic pop and rock has over 1600, which could make difficult to have high precision and recall of minority classes.

4 Model development

The strategy to develop the model followed the next steps: Data manipulation, feature engineer and feature selection are used to create several datasets. Later, algorithm benchmark is performed over these data sets and finally some hyper-parameter tunning is executed over best performing algorithms.

4.2 Data manipulation

Several procedures were tried and combined, creating different datasets. Most of these procedures were performed to a set of features, however, oversampling was applied to the instances. Oversampling was performed by *boostapping* each class until reach same number of samples for each class in an attempt to compensate for the imbalance in the classes distribution.

4.3 Feature engineer

Textual data was transformed into a vectors for each unique word present in the dataset tags. However, the majority of the most frequent words are meaningless as they are extremely common as they appear in most song tags, such as pronouns, prepositions, conjunctions and interjections.

Two strategies can be used to solve this issue: The first method is to ignore those words using a list of "stopwords" as reference. Such list is generic and contains most common prepositions, pronouns and conjunctions in English. However, it does not deal with domain specific frequent words or multilingual dataset. Therefore, a second strategy is to use: Term Frequency Inverse Document Frequency (TF-IDF) which counts the frequency of words for each instance (Term frequency) but also weight it with inverse value of the count of instances a word appears (Document Frequency). Then, if a word is shown in many instances its predictive power is less important. Nevertheless, the lyrics have been already pre-process, as summarized unique keywords were given in the dataset. Also, some words were so infrequent that can be uninformative or lead to overfitting. Those were ignored over a threshold indicating that the word should appear at least in ten instances to be considered as a vector. Otherwise those words could be used for certain algorithms as *Decision trees* and overfit the model (R. Baeza-Yates and B. Ribeiro-Neto 2011).

A separated matrix was created for titles following the same procedure.

To test how useful TF-IDF is over simple frequency counts, both resulting features were tested with *mutual information*, more in about it in feature selection.

Additionally, a numerical feature was calculated by counting the number of tags of each song. It is an interesting feature as each music genre has different distributions of tags count.

Categorical attributes are presented in the song *key*, to handle such attributes one-hot encoding was performed.

Numerical data was normalized. Several tests were applied to intend to stablish the optimal normalization strategy. However, no feature shown gaussian distribution to use Standardization or proof of power law distribution for Power transformation, therefore simple normalization was implemented. Also, discretization was tried.

4.4 Feature selection

The dataset changed from 156 features to over 5000 features. As many of these features may contain misleading information, feature selection was performed to select the most informative ones. To such task, *Mutual information* was calculated for each feature, as is already implemented in *Sci-kit learn* package. Several dataset were created:

- Metadata features + text vectors (B+NLP)
- Only text vectors (NLP)
- Metadata features + unknown vectors (B+Vec)
- Only unknown vectors (Vec)
- Full dataset with oversampling (Over)
- Best 2000 features with oversampling (Over2k)
- Only best 1000, 2000, 3000 features, respectably (X1k, X2k, X3k).
- Full data set (Full)
- Full dataset with numerical data with uniform discretization and bin 20 and 50 (bin20, bin50)

4.5 Algorithm selection

Some intuitions about the characteristic of the dataset were extracted by exploring the dataset. Those intuition were tested by systematically applying off-the-shelf algorithms provided by *Sci-kit learn*, for each of the resulting datasets. The algorithm selection considerer those commonly used in the literature. The algorithms are: *Decision tree* (DT), *k-NN* (KNN), *Logistic Regression* (LR), *Naïve Bayes* (NB), *Random Forest* (RF) and *Support Vector Machines* (SVM).

Metrics of Accuracy and F1-score were stored in Table 1. As the problem is multiclass and imbalanced it is necessary to measure how each class perform, to accomplish such task and average of F1-score for each class was register.

	DT	**KNN**	**LR**	**NB**	**RF**	**SVM**	**1-R**
B+NLP	0.38	0.19	0.57	0.41	0.6	0.53	0.26
NLP	0.37	0.16	0.56	0.41	0.58	0.34	**0.27**
B+Vec	0.33	0.32	0.48	0.41	0.47	0.51	0.25
Vec	0.33	**0.36**	0.45	0.40	0.44	0.50	0.25
Over	**0.43**	0.30	0.61	0.43	**0.62**	**0.66**	0.24
Over 2k	0.42	0.32	0.62	0.46	**0.62**	**0.66**	0.24
X1k	0.40	0.22	**0.63**	**0.46**	0.61	0.64	0.25
X2k	0.42	0.24	0.62	0.42	0.61	0.65	0.25
X3k	0.4	0.25	0.62	0.4	0.60	0.64	0.25
Full	0.41	0.19	0.62	0.41	0.60	0.64	0.25
bin20	0.32	0.35	0.51	0.38	0.45	0.51	0.24
bin50	0.31	0.35	0.52	0.42	0.50	0.54	0.24

Table 1-Accuracy scores for each dataset and algorithm. Highest score for each algorithm in bold. Full table including F1-scores in appendix.

Baseline of one-R was selected as baseline because it allows to compare how the aggregation of features compares using only the best one. It has an accuracy score of 25% with the full dataset.

4.6 Hyper-parameter tunning

Tuning hyper-parameters can have a modest impact in the accuracy of most models in many cases. However, the risk of overfitting is high. Therefore, to reduce such risk the systematic search for best parameters, a cross-validation within the training dataset was used.

As trying several possible combinations of parameters is a time expensive task, only the best performing vanilla algorithms were tuned. In these case RF, SVM and LR.

5 Evaluation and Analysis

As expected from the data exploration, *Naïve Bayes* was impacted negatively from the high dimensionality and correlation among features. Its scores improve with fewer features.

High dimensionality impacted *Decision trees* as well, making it quite complex and less capable to generalize. Scores could improve by limiting the maximum depth of the tree.

k-NN performed poorly, even worse than the baseline in most cases. However, is quite sensitive to parameter selection. In this case the parameters k=1 and distance metric = 'Euclidian' were tried following literature examples.

Logistic Regression in the other hand, was capable of balance features assigning weights to each feature. Using L1 norm penalty could improve its performance as it allows the algorithm to assign weight zero to unimportant features, performing itself some feature selection. However, it did not

converge to a solution, indicating that solution could be non-lineal.

Neural networks were then an ideal candidate as it could find some non-lineal relation, however, it was not tried because much more work is required to design a multilayer architecture and it is likely that much more training data is required as well.

SVM can overcome the non-linearity of the solution as well, in fact, using *polynomial* kernel was slightly better than LR in most cases.

Random forest was used as stacking technique of DTs and improve greatly its performance by creating trees with different combinations of features. This could mean, there is room for a better selection of features that could increase greatly de accuracy and F-1 scores.

Oversampling increased the performance of most of the algorithms, combined with feature selection (best 2000) had the highest average score among the different datasets.

The inclusion of text features (NLP) also increased the performance of most algorithms when compared to the result of only using audio vectors and metadata.

Binning did not help. It was used as a technique to help to reduce noise of continuous features. More experimentation is required.

6 Conclusions

Feature selection is an important component of Machine learning model development as it can account for differences up to 16% in accuracy on average.

Oversampling and the inclusion of TF-IDF texts features have the greatest impact among all the data manipulation techniques.

Stacking methods could increase performance as Random forest increased greatly the performance of DT, probably by creating combination of features that can be similar to the *Temporal Integration* and *domain* features reviewed in literature (A. Schindler and A. Rauber 2012) were statistical summaries of features were used by selecting a number previously known vectors.

Techniques that may help to produce such summary features are reduction of dimensionality as Principal Component Analysis (PCA) that capture the main variation of features, However, the use of such techniques were out of the scope of this paper.

References

T. Bertin-Mahieux, D. P.W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR), 2011.

A. Schindler and A. Rauber. Capturing the temporal domain in Echonest Features for improved classification effectiveness. In Proceedings of the 10th International Workshop on Adaptive Multimedia Retrieval (AMR) , 2012

Li, T., Ogihara, M. and Li, Q., 2003. A comparative study on content-based music genre classification. Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval - SIGIR '03,.

Diab, O., Mainero, A., & Watson, R. 2012. Musical Genre Tag Classification With Curated and Crowdsourced Datasets.

R. Baeza-Yates and B. Ribeiro-Neto 2011. Modern Information Retrieval. Addison Wesley, pp. 68-74.

T. Mitchell, 1997. Machine Learning. Chapter 3: Decision Tree Learning.

Tan, P., Steinbach, M., Karpatne, A. and Kumar, V., 2005. Introduction To Data Mining. Pearson.