COMP 551 Applied Machine Learning Winter 2020
Classification of Textual Data

Smith, Brandon
McGill University
brandon.smith@mail.mcgill.ca

Dawood, Sully
McGill University
sully.dawood@mail.mcgill.ca

Yang, Danning
McGill University
ydn2012@hotmail.com

I. Abstract

This project focuses on getting an idea of the performances of five different classifiers on two different data sets: 20 News Groups; and IMDB Reviews. We first pre-processed the data with Scikit-Learn's count vectorizer, standard scaler, and normalizer, then construct the 5 different classifications models: Logistic regression, Decision trees, Support vector machines, Ada boost and Random forest. We used grid search cross validation strategy to tune the count vectorizers, and then subsequently to tune the models' hyper parameters. This resulted in the group finding the optimal parameters of best performance of each model-data set pair. Overall,the performance of each model increased between 3% - 15% when compared with the default parameters of each respective model. The model with best performance with data set 20 news group is logistic regression classifier , with an accuracy of 68%, and the model with best performance with data set IMDB reviews is logistic regression and linear SVC, with an accuracy of 87%. Combined together the overall accuracy achieved was 76.5%

II. Introduction

The primary goal of the paper is to attempt to train 5 different classification models on the two data-sets: 20 News Groups; and IMDB Reviews. The steps to achieve said goal are to pre-process the raw text data using Scikit-Learn's Count Vectorizer, scale and normalize the now numerical data using Scikit-Learn's Standard Scaler and Normalizer, and lastly fit one of the following models in Scikit-Learn's library to the prepared data: Logistic Regression; Decision Tree Classifier; Support Vector Classifier; Ada Boost Classifier; and Random Forest Classifier. Scikit-Learn's GridSearchCV helped with separately tuning the pre-processing step and the classifiers' Hyper Parameters. By splitting up the search into two separate sections the running times of the code was able to be drastically reduced for minimal hits to accuracy.

Both data-sets are extremely large, Twenty News Groups with 18828 text documents,11314 for training, 7532 for testing, and IMDB Reviews with 50000 text documents, 25000 for training, 25000 for testing. The 20 News Groups has 20 categories (classes), whereas IMDB Reviews only contains positive and negative classes. Unsurprisingly, the IMDB Reviews has an average accuracy much higher than 20 News Groups since it has less classes. The best performing model on 20 News Groups is Logistic Regression and the best for IMDB Reviews is a tie between Logistic Regression and Support Vector, which demonstrates the versatility a simple model such as Logistic Regression can have. Additionally it shows that the binary nature of the IMDB Reviews data-set lended well to SVC because of it's binary nature.

III. Related work

Working with text in machine learning at first may seem like a daunting task because of the infinite possibilities for what the text might contain, but through techniques such as Stemming, Lemmatizing, and most importantly through pattern matching, you can drastically reduce the set of your input text. Further, you

then can apply the technique implemented in Scikit-Learn's CountVectorizer, which essentially performs operations on the text to simplify it, and then counts the occurrences of each word. From that point, further techniques may be applied such as modifying the counts based on frequencies of the words, and furthermore scaling and normalizing the data. Through these methods (among many others) text data can be reduced to a set of numerical data which classifiers can then easily operate on.

Multi-class Classification is difficult because by nature classification models fit a binary classification boundary where either side of the boundary is 0 or 1. The core idea behind solving this problem is that any multi-class classification can be reduced down to a series of binary classifications working in unison. The two main strategies for this are "One vs Rest (ovr)" and "One vs One (ovo)", where as the name suggests ovr classifies a binary boundary of being One or all the rest, and ovo classifies a binary boundary of being one or another one. Once in binary form they can be easily passed through models such as trees or support vector machines.

IV. Dataset and setup

Both data-sets are in text format, so as mentioned in the Introduction, we first must pass the data through a Count Vectorizer to transform the data into numerical feature vectors so that the classifiers can understand the data. Then we use a Scaler to center the data around a set of statistical points (be it the mean, or minimizing the std-dev, etc). Once we've scaled our data around a chosen value, we want to further normalize the data so that the bounds of the data are within a range that works best for the classifiers. In Scikit-Learn most of the available pre-processing options come packaged within the CountVectorizer class, so using the Scikit-Learn Default models, we performed a GridSearchCV on the CountVectorizer parameters to find a good estimate on the ideal pre-processing to pass as parameters to the CountVectorizer when we want to tune our classifiers. The following table are the most relevant parameters that were set for each data-model pair.

Best parameter of Count Vectorizer					
parameters	lowercase	max df	max features	min df	ngram range
20news &LG	True	0.3	100000	3	(1,1)
IMDB &LG	True	0.4	10000	3	(1,3)
20News &DT	True	0.4	10000	2	(1,1)
IMDB &DT	True	0.4	10000	3	(1,3)
20News &LSVC	True	0.3	100000	2	(1,2)
IMDB &LSVC	True	0.5	10000	3	(1,3)
20News &AB	True	0.5	10000	2	(1,1)
IMDB &AB	True	0.4	10000	3	(1,3)
20News &RF	True	0.5	100000	2	(1,2)
IMDB &RF	True	0.5	100000	2	(1,2)

Additionally the data could have been pre-processed using more aggressive approaches to reduce the number of features, however for the purposes of this paper we decided to use only the methods provided within Scikit-Learn 0.22

V. Proposed approach

The first model used was Logistic Regression (or logit model), which is the probability of a certain class or event existing. This is a binary existence (e.g. True/False) which can be extended to model several classes of events. The second model utilized was the Decision tree, a powerful and popular tool for classification and prediction. A Decision tree is a hierarchical node-like

structure, where "tests" are conducted on internal nodes and each branch represents an outcome of the test. Lastly, as progress is made through the branches, the class labels are determined from the leafs. The third model studied was Support Vector Machines (SVM), which is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. The Random Forest Classifier is another useful classifier to prevent over fitting; it uses the number of decision tree classifiers on various sub-sects of the data set and uses averaging to improve the predictive accuracy and control over-fitting. Lastly, there was also the utilization of the Adaptive Boosting model (AdaBoost), where the output of other machine learning algorithms is combined into a sum that is weighted, which is subsequently used in classification.

To be able to get the best parameters, we use grid Search Cross Validation strategy with the CountVectorizer being pre-determined (see section IV). The simplest way is to put all possible parameters together, then run one search with 5 to 10 cross validation. However due to the huge amount of possible parameters, this will take a huge amount of time to run(few days maybe), so one of the methods we tried was to test with 1 input only, with maximum 5 different parameters. First start with large scale, then narrow down till we find the exact best parameter. Ex: to find the best parameter: max_depth, we first do search on max_depth, with parameters [10,100,1000], gives the best which is 100, then set with parameter [50,100,200,300,500], gives the best which is 300, then set the parameter which is [260,280,300,320,340], this still gives the best parameter = 300, so we finally decide the best parameter of max_depth = 300. After find the one best parameter, we move to the next parameter and repeat the process. Another method used was to reduce the input size and run the massive cross-validation on all parameters, to reduce the set of parameters to a subset of the original, and then increase the input size. A final method we used was Randomized Search to fix the total number of iterations to a constant number.

VI. Results

Figure 1: *Twenty News Groups Results*

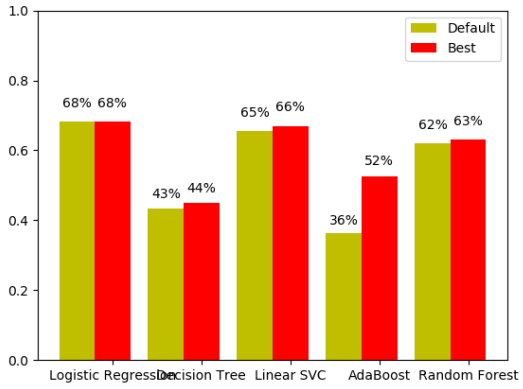
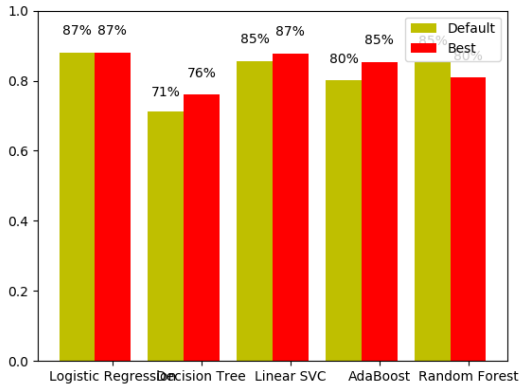


Figure 2: *IMDB Reviews Results*



Logistic Regression - Twenty News Groups: parameters(class_weight = 'balanced', intercept_scaling = 10, max_iter = 10000, tol = 0.00001, solver = 'sag')
The default and the above hyper-parameters both achieved an accuracy of 68%
Logistic Regression - IMDB Reviews: parameters(class_weight='balanced', max_iter=1000, solver='sag', tol=0.001)

Same As above, defualt and above hyper-parameters both achieved 87%

Decision Tree - Twenty News Groups: parameters(criterion = "gini", max_depth = 300, min_samples_split = 140, max_leaf_nodes = 2300, ccp_alpha = 0.000001)
The decision tree classifier does not gives a good performance on 20 news data group, after doing several different grid-search cross Validation process, with different hyper-parameters, the accuracy of best parameter in the grid search has only went up to 47%, by refitting the training data into a new decision classifier with those parameters, and predict the testing data, the accuracy is 43%, which is 3% higher compare with the default parameter(40%).
Decision Tree - IMDB Reviews: parameters(class_weight='balanced', max_leaf_nodes=100, min_samples_leaf=10, min_samples_split=0.01, splitter='best')
For the IMDB reviews, the performance is much better, by doing the same GSCV process as above, after refit the training data and predict the testing data with best parameter, the accuracy is 76%, compare to 71% with default parameter, overall the decision tree works better with less class, and due to the small number of class, we find out that it requires also smaller number of max_depth, with higher number of min_samples_split, which gives a smaller running time.

Linear SVC - Twenty News Groups: parameters(C=0.1, dual=False, fit_intercept=False, intercept_scaling=10.0, max_iter=100, tol=0.001)
Linear SVC does perform a pretty good result also on 20 news groups, with a 66% but already a 65% before tuning.
Linear SVC - IMDB Reviews: parameters(C=0.1, fit_intercept=False, tol=0.001)
Linear SVC for IMDB has become the same performance as Logistic regression after tuning, which shows that this is also a good model to use for 2 classses data.

Ada Boost - Twenty News Groups: parameters(base_estimator = DecisionTreeClassifier(criterion = "gini", max_depth = 300, min_samples_split = 140, max_leaf_nodes = 2300, ccp_alpha = 0.000001), learning_rate = 0.01, n_estimators = 100)
Before doing grid search cross validation, both of them has use the same decision tree classifier with the parameter above as the base estimator. This has already slightly increase the accuracy since it gives better performance than the default base estimator(decision tree with max_depth = 1), however this also slightly slows down the running time. For the 20 news group, the best parameters are(learning_rate = 0.01, n_estimators = 100), with the performance of 47% with predicting the testing data, compare to the default parameter with only 35%.
Ada Boost - IMDB Reviews: parameters(base_estimator = DecisionTreeClassifier(max_depth = 17, min_samples_split = 560, ccp_alpha = 0.0001, min_samples_leaf = 12), learning_rate = 0.1, n_estimators = 100)
the best parameters result in a performance of 47% with predicting the testing data, compare to the default parameter with only 35%. Overall, we find out that by increasing the number of estimator will increase the accuracy but also increase the running time. The running time of this model is much much higher compare with the others. Overall this classifier is still not a good one for a data with multiple classes.

Random Forest - Twenty News Groups: parameters(criterion = "gini", max_depth = 300, min_samples_split = 140, max_leaf_nodes = 2300, ccp_alpha = 0.000001)
Random forest has the exact same best hyper parameter than decision tree, however it's performance is much better than decision tree, with the same hyper parameter, the accuracy of testing data 20 news group is 66%, which was 61% with default.
Random Forest - IMDB Reviews: parameters(ccp_alpha=1e-05, class_weight='balanced', criterion='entropy', max_features='sqrt', min_samples_leaf=0.01,

min_samples_split=10, oob_score=True) imdb review, the best parameter is 85%, and with random forest strategy(due to the lack of time), we didn't find one with better performance.

VII. Discussion and Conclusion

The result shows that logistic regression model and Linear SVC works better with classifying multi class data. And the performance with all be better with less classes a data have Future investigation: We believe that we can do a linear regression on IMDB reviews, since each txt file had a numerical rating form 1 to 10 attach to it, in this case I believe linear regression will give a better performance.

VIII. State of contribution

All members participated actively on all sections of the project.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC.

[2] Kevin P. Murphy. *Machine Learning : A Probabilistic Perspective*. MIT Press.

[3] Andrew L. Maas et al. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[4] *20 News Groups*. Irvine, CA 92697-3425, Sept. 1999. URL: http://qwone.com/~jason/20Newsgroups/.

[5] *Scikit-Learn*. URL: https://scikit-learn.org/stable/.

[6] Various. *Towards Data Science*. URL: https://towardsdatascience.com/.