Performance analysis of Five Models on Twenty Newsgroups Datasets and IMDb Reviews

Angela Fan, George Lungoci, Hanying Shao

Abstract

In this project, we analyzed the performance of five models respectively upon 20 newsgroups datasets and the IMDb movie reviews, all in text documents format. Two of the models got our attention: the logistic regression and the support vector machine (SVM). Both are supervised machine learning algorithms for which several factors were tested to compare the performance. According to the paper from *Axum Technologies lab*[1], SVM works well with unstructured and semi-structured data like text and images while logistic regression's performance is "better with already identified independent variables", which is what we successfully showed in our experiment. We also tried to demonstrate that the risk of overfitting is relatively low in SVM, therefore being the model that constantly displays the best accuracies. Logistic regression was shown to perform better than random forest classifier when increasing the variance, a result supported by Kaitlin Kirasich, Trace Smith and Bivin Sadler, three data scientists, in their paper[2] (2018). AdaBoost, instead of being a classifier itself, should be applied to other classification algorithms, specifically to those that perform poorly, in order to get higher accuracy, which is why it acted poorly when we applied directly to the datasets. We compared the accuracies before tuning (using models' respective default hyperparameters) and after tuning for the five methods, where the accuracies were seen to be stable, a phenomenon that could be explained by the fact that the computer chose the same parameters as the default ones. Otherwise, in overall, SVM had the best accuracy with 0.80 for 20 newsgroups and 0.89 for IMDb while decision tree classifier had the weakest performance (around 0.5), as expected, mainly due to the fact that it is a method very sensitive to small perturbations and overfits easily.

Introduction

The five models that we used to test our datasets are: logistic regression, decision trees, support vector machines (that we will abbreviate as SVM), Ada-boost and random forest.

The logistic regression model uses an S-shaped curve that takes a real-valued number to map to value between 0 and 1, while the SVM creates a line or a hyperplane that separates data into classes. Therefore, SVM presented better accuracy, which is why we added a new feature for the logistic regression to improve its accuracy, notably using different decision boundaries with distinct weights near the optimal point.

Ada-boost, like random forest classifier, is an ensemble classifier, more specifically, it is made up of multiple classifier algorithms where the output is the prediction based on the result of the different models. However, what is different from the two models is that the random forest classifier uses the bagging method, which implies that it trains the individual models in a parallel way, while Ada-boost learns from previous mistakes such as misclassified data points and improves the performance by increasing the weights of these points. Therefore, at the beginning of the project, we formulated the hypothesis that Ada-boost will give better accuracy; however, of a little bit of surprise, we found that Ada-boost was the weakest model, followed by decision trees and random forest model (differ by 1%).

[1] *Logistic Regression Vs Support Vector Machines (SVM),* Patricia Bassey

[2] *Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets,* Southern Methodist University

The reason for the poor performance of decision trees can be explained by the fact that it is a model built on the entire dataset whereas the random forest builds multiple decision trees from randomly selected features or variables of the dataset and get the average results from it. However, after tuning, the accuracy of random forests increased by a lot, more specifically by 17%, the reason presented in the result section.

In summary, from decreasing order of accuracy, we have SVM (having the best performance), logistic regression, random forest, decision trees and finally Ada-boost model (low accuracy).

Related work

Several reports were published by researchers of Columbia University on comparisons of different classification models, the most relatable being the one between logistic regression and SVM[3] (2012). In the paper, the author talked about how logistic regression model was widely used in machine learning and applied statistics, but recently (referring to 2012), SVM became a good alternative. Applying the kernel function, SVM separates data in a more logical way, therefore displaying better accuracy. In addition, logistic loss does not go to zero even if the point is classified sufficiently confidently, which also leads to accuracy inefficiency. Therefore, the conclusion from this report is that "SVM performs marginally better than logistic regression", which is proved throughout our experiments.

Another useful work was about random forest and logistic regression (LR), as mentioned in the abstract, a very recent paper (2018). It implies that for the cases of more complex datasets, linear-based algorithms may give less accurate results, so the author predicted that random forest model would demonstrate better performance. However, as we increase the variance, logistic regression classifier performs higher overall accuracy.

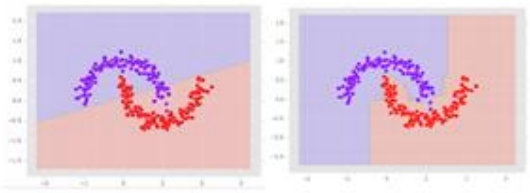

Figure 1: decision boundary with complex data structures, LR on the left and random forest on right

Another paper that we found useful, written in 2005 by Guy Leshem[4], talked about Ada-boost not being a classifier by itself, but should be combined with other classification models in order to have better performance. Therefore, combining random forest and Ada-boosting to get stronger and more accurate learner is a common way of doing, says Leshem. Also, "the error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them", but we also know that using random selection of features would yield unpredicted errors.

Finally, we looked at a publication[5] (2012) where the authors compared the classification results obtained from random forest and decision trees on 20 data sets of different size, collected from UCI repository, very similar to what we are doing. To avoid over-fitting, accuracies were obtained using 10-fold cross validation, and it was concluded that random forest has "increased classification performance and yields more accurate and precise results in the cases of large number of instances", which confirmed our previous hypothesis and answered our doubts.

Dataset and setup

The 20 newsgroups dataset that we are using to test our code was originally collected by Ken Lang; it contains nearly 19,000 newsgroups posts on 20 topics that are split into two subsections: training and testing. We used the 'scikit-learn' tools to analyze this collection of text documents, where the first step was to get the bag of

[3] *Comparison between SVM and Logistic Regression: Which One is Better to Discriminate*, Diego Alejandro Salazar, Jorge Iván Vélez and Juan Carlos Salazar

[4] *Improvement of Adaboost Algorithm by using Random Forests as Weak Learner and using this algorithm as statistics machine learning for traffic flow prediction*, Guy Leshem

[5] *Random Forests and Decision Trees*, Jehad Ali, Rehanullah Khan, Nasir Ahmad, Imran Maqsood

words in order to get the occurrence of each one from it. We then built a pipeline to assemble the steps that can be cross-validated together using different parameters, specifically from vectorizer to transformer to classifier. We extracted the features by respective categories, such as atheism, politics, sport, etc., and as a way to deal with occurrences, we used the tokenizing to filter stopwords. To avoid having abnormally high average count of words in longer documents, we preprocessed the dataset using the tf-idf command. However, to have a more visual comparison, we decided to add the new feature of using tf as well, where idf, referring to 'inverse document frequency', was set to false.

The second set was the IMDb reviews, a very well-known website where one can see reviews and comments on any movie. Collected in 2011, the dataset contains a set of 25,000 movies respectively for training and testing. Here, we are working on highly polar movies, therefore the reviews are either positive or negative, ignoring all in-between comments. Each word was considered as a feature, where we looked at the frequency of all features and restricted our running dataset to be those having frequencies higher than 1 (basically meaning that we used all the non-empty words). The dataset was stored in a kind of matrix where each row represented a review and the columns were the words. The more features/words we have, the better the accuracy would be.

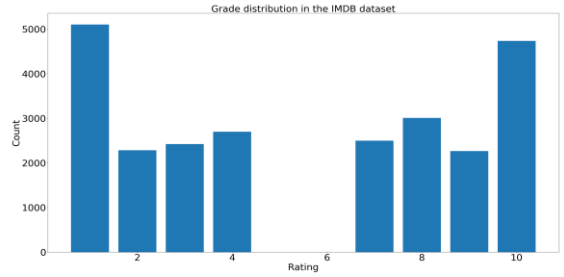

Figure 2: reviews grades distribution for IMDb dataset

Proposed approach

We used the pipeline utility and a for loop to test on the models one by one by changing only the 'clf' parameter. We also used the built-in cross validation function to get the mean CV score and the standard deviation. To train the best hyperparameter, several arguments were being used. We first used a 5-fold cross-validation to test our code; however, as the paper (2012) on the comparison of random forests and decision trees implies that a 10-fold would be a better choice, we changed our code and indeed got a great increase in most of the performance of the models.

In addition, for each model, we considered appropriate parameters to test the best cv score, for example for the support vector machine model, we considered the penalty, where both l1 and l2 were tested to deduce the better performance. In contrast, for Ada-boost, instead, we tried various learning rates (between 0 and 1, equally spaced using np.linspace) since it would not make sense to test penalty on it.

We used Grid Search including the pipeline to test all possible combinations of possible values for different parameters to find the combination that gives the best accuracy and CV score. We also plot the graph of the CV scores for each value we are testing to visualize how the parameter affects the result. The standard deviation of the best CV score is also identified in the graph by the redlines in the result section.

Besides the new features we mentioned earlier, we also tried to improve the accuracy by doing multiple parameter tuning, as well as using different preprocessing approaches. Instead of using 'tf-idf', we tested by only applying 'tf' to see the relative performance.

A very useful previous work we consulted and referred to is the one on *Kaggle*, where we implemented the pipeline after learning from it. Another very helpful website we used was *scikit learn*[6], where installation instructions and grid search strategy were very well explained, as well as feature vectors suitable for machine learning.

We added two new models to test our code as well: multinomial naïve Bayes and multilayer perceptron. We chose the naïve Bayes model to respond mini project 1, while multilayer perceptron is known to work well with

[6] https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

nonlinear functions, as well as complex ones, as it is a neural network. We are expecting higher accuracy from it, and a better hyperparameter.

For decision trees model, the parameter we are most interested in is the maximum depth since it is all about splitting the dataset into subsets, in binary, until there is no more possible partitioning. It learns rapidly through tuning.

The random forest classifier, as mentioned earlier, is an ensemble learning method; in order to get more accurate results, we used the largest number of parameters to test the best hyperparameter from it.

The logistic regression, being the most common model, is efficient to train, but overfits easily with smaller datasets, which was not the case here. Therefore, to get higher performance, we increased the maximum iteration to a relatively large number.

Results

With default hyperparameters, for the 20 newsgroups dataset, the best accuracy achieved was 0.7991 by the support vector machines, with a relatively high cross validation score mean of 0.75. In opposition, Ada boost had the lowest performance with 0.47. However, we did not see a very big increase in terms of performance before and after tuning, which may be explained by the fact that our machine learning implementation chose the same parameters as the default ones.

Overall, the accuracies stayed stable. However, for random forest model, it improved for both dataset, 17% for 20 newsgroups and 3% for the IMDb one. One way to explain it would be that from tuning, misplaced and irrelevant data were degraded and the selected features used to build decision trees from it got improved considerably. Even if it is not the best model to run on our datasets, we noticed that training random forest classifier is very efficient and effective.

Support vector machine uses a data-set driven approach to find the standard deviation; we see from Figure 4 that it extends to a much larger distance, which means that the values are spread out, the variance being high, thus more noisy than smaller s.d. We can also read the best cv score, marked in red, relative to each parameter on the graph, for example in Figure 4, we refer to the C parameter (equals to 0.77), the penalty parameter of the error term, of SVM classifier of the 20 newsgroups dataset. The standard deviation is particularly useful to the hyperparameter training, where the larger we get, the better the accuracy we would expect from it.

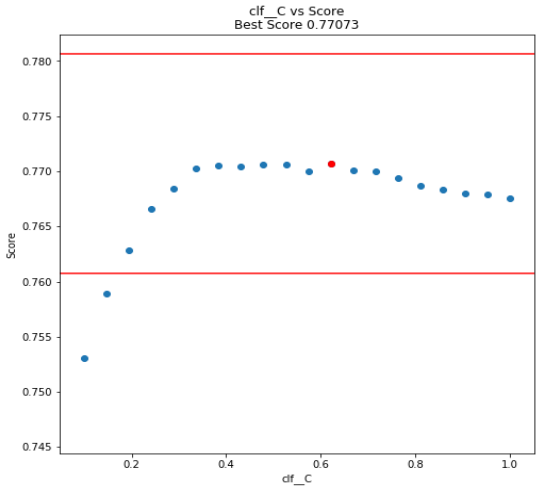

Figure 3: cv score and standard deviation for 20 newsgroups dataset, SVM classifier

The best accuracy for both datasets is achieved by the support vector machine, closely followed by the logistic regression after tuning, differ by less than 1%. The worst model being Ada-boost. However, the multilayer perceptron gave an even higher accuracy, with a relatively high cv score as well (see Figure 4), thanks to the fact that it deals nonlinear functions better than the other models.

20 Newsgroups Dataset:	MultinomialNB	MultiLayer Perception	Gradient Boosting
Accuracy	0.7088	0.8117	0.7177
CV score mean	0.6244	0.7549	0.6273
CV score standard deviation	0.00721	0.0165	0.0082

Figure 4: accuracy for the two extra models testing on 20 newsgroups dataset

20 Newsgroups Dataset:	Logistic Regression	Decision Trees	Support Vector Machines	Ada Boost	Random Forest
Accuracy before tuning (with default	0.7811	0.4967	0.7991	0.4749	0.5027
Parameters that give the best result	'C': 100, 'penalty': 'l2'	'criterion': 'gini', 'max_depth': 1000	'C': 0.6211, 'loss': 'squared_hinge'	'learning rate': 0.7631, 'n_estimator': 50	'criterion': 'gini', 'n_estimator' : 50
Accuracy after tuning	0.7898	0.4944	0.8041	0.4794	0.6739
Best CV score after tuning	0.763	0.4461	0.7707	0.4039	0.6144
IMDB Dataset:	**Logistic Regression**	**Decision Trees**	**Support Vector Machines**	**Ada Boost**	**Random Forest**
Accuracy before tuning (with default hyperparameter)	0.8831	0.703	0.8772	0.8034	0.7367
Parameters that give the best result	'C': 10, 'penalty': 'l2'	'criterion': 'entropy', 'max_depth': 10	'C': 0.3368, 'penalty': 'l2'	'learning rate': 0.7157, 'n_estimator': 50	'criterion': 'gini', 'max_depth': None
Accuracy after tuning	0.8812	0.7092	0.8842	0.8036	0.7697
Best CV score after tuning	0.8954	0.7114	0.8969	0.8059	0.7848

Figure 5: result compilation table for both datasets, in blue being the increased accuracies, in red, the decreased ones

A possible reason for the decrease of the accuracy may be that the tuning of the hyperparameters does not contribute to the learning because of the randomization in the fitting. It lowers the performance of the model, for example for the decision tree, by chance.

Compared to using tf-idf, the accuracies in overall lowered by around 1-2%, which means that the method one uses to pre-process a dataset is indeed important.

20 Newsgroups Dataset using 'tf'	Logistic Regression	Decision Trees	Support Vector Machines	Ada Boost	Random Forest
Accuracy	0.6771	0.5098	0.7725	0.468	0.4982
CV score mean	0.5749	0.4274	0.7081	0.3951	0.4526
CV score standard deviation	0.0057	0.0031	0.0057	0.0097	0.0037

Figure 6: accuracy using 'tf'

Discussion and Conclusion

All hypothesis and expected results have been confirmed by our experiment. For the IMDb dataset, in order to get higher accuracy, we counted all words that have frequencies higher than 1 to be a feature. For both datasets, the support vector machines classifier displayed better performance, followed very closely by the logistic regression model.

Combining Ada boost with random forest as stated in one of the related works we talked earlier, the accuracy did not improve by a lot, which contradicted our expectation. We suspected some errors in the coding, therefore would like to try implementing it again in the future, as well as trying to combine with other poorly performed classifiers and compare if possible.

We learnt k-nearest neighbour (kNN) since the beginning of the class, but have never used it in real life. Also, recognizing images instead of text or numerical data may be an interesting thing to do during spare time, which could be another possible future investigation proposal. Machine learning consists of applying all the theory to practice; it can already answer in-depth statistical questions, where it would be nice if all applications can be expanded to multi-labeled datasets instead of binary classification (for example what we did for IMDb), therefore we may improve our implementations to test more variant variables.

Statement of Contributions

George Cristian Lungoci worked on establishing the data processing, performing the experiments and improving the overall code.

Angela Fan contributed to the new features, as well as writing the report.

Hanying Shao is responsible for building up the implementation code and running experiments on accuracy of the models.

References

Ali, Jehad & Khan, Rehanullah & Ahmad, Nasir & Maqsood, Imran. (2012). "Random Forests and Decision Trees". International Journal of Computer Science Issues (IJCSI), page 9.

Axum Labs. Research lab for axum technologies Ltd. (2019, September 19) "Logistic Regression Vs Support Vector Machines (SVM)". Retrieved from https://medium.com/axum-labs/logistic-regression-vs-support-vector-machines-svm-c335610a3d16#:~:text=Difference%20between%20SVM%20and%20Logistic%20Regression&text=SVM%20is%20based%20on%20geometrical,regression%20is%20vulnerable%20to%20overfitting.

Diego Alejandro Salazar, Jorge Iván Vélez and Juan Carlos Salazar. (2012). "Comparison between SVM and Logistic Regression: Which One is Better to Discriminate". Volume 35.

Drakos, Georgios. (2018). "Support Vector Machine vs Logistic Regression". Retrieved from https://meium.com/axum-labs/logistic-regression-vs-support-vector-machines-svm c335610a3d16#:~:text=Difference%20between%20SVM%20and%20Logistic%20Regression&text=SVM%20is%20based%20on%20geometrical,regression%20is%20vulnerable%20to%20overfitting.

Kirasich Kaitlin, Trace Smith and Bivin Sadler. (2018). "Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets". Volume 1, Article 9.

Leshem, Guy. (2005, January 25). "Improvement of Adaboost Algorithm by using Random Forests as Weak Learner and using this algorithm as statistics machine learning for traffic flow prediction". Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.109.224&rep=rep1&type=pdf

Maas, Andrew L., Daly, Raymond E., Pham, Peter T., Huang, Dan, Ng, Andrew Y. and Potts, Christopher. (June 2011). "Learning Word Vectors for Sentiment Analysis". Retrived from http://www.aclweb.org/anthology/P11-1015

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830