COMP551 Project 2: Classification of Textual Data

Agnes Liu **Si Yi Li** **Matteo Cacciola**

Abstract

Multi-class classification is widely applied when dealing with textual datsets due to the high complexity of text documents. With tuning of hyperparameters, this project compared performances of different models in assigning class labels according to textual inputs. We found that in multi-class classification with less samples per class, it is common for learning models to suffer from overfitting. In binary classification with large number of samples per class, the tested models in general received better performances. SVM achieved the best prediction accuracy, comparing with Logistic, Decision Trees, Ada Boost and Random Forest. All our code are accessible in our Google Drive.

1 Introduction

Multi-class text classification is an ambitious yet common approach in machine learning which assigns multi-class labels to textual data. To gain insights of this type of classification while boosting our understanding towards some applicable learning algorithms, we compared the performances of the following machine learning classifiers: Logistic regression (LR), Decision Trees (DT), Support Vector Machines (SVM), Ada Boost (AB) with decision stump, and Random Forest (RF).

We used 2 datasets for the cross comparison of these models. In particular, 20 news groups Dataset from Sci-Kit Learn (Pedregosa et al., 2011) contains 20 class labels and Large Movie Review Dataset from Stanford University (Maas et al., 2011) includes only 2 class labels.

Our experiments illutrated that, under the best settings of hyperparameters and feature selection methods elected by our run time data, all models performed rather poorly (around 40-64%) in the 20 news group dataset, where they faced the challenge of multi-class classification. On the other hand, the overall model performance for IMDB was way better(in the range of 72-90%), suggesting the number of classes and samples per class could be influencing factors for model prediction accuracy. Importantly, we noticed that despite the unstable performance of AdaBoost, it did not suffer from overfitting. In both datasets, SVM displayed the overall best performance amongst all classifiers.

2 Related Work

Training a machine to learn from textual data is significantly more challenging than the equivalence on numeric data. The fact that texts are usually rich in vocabulary gives rise to large dimensions of the feature space, which likely triggers the problem of overfitting. In addition, models cannot recognize newly encountered words or tokens as features if it has not learnt it from the training process and vice versa. That is, the redundancy of human languages adds up to the overfitting issue. Moreover, learning from text is usually attempted in a multi-class classification manner, as texts can convey a lot more than binary information. A study from Li et al. cross comparing multi-class classifiers have shown that as number of distinct class increases, prediction accuracy drops drastically, with no model significantly beats the others. (Tao Li and Ogihara, 2014) These obstacles brought by such problem setting create more constraints over data preprocessing and feature representation. We filtered features to reduce dimension, and represented them with the method of Bag of Words (to be discussed in section 3.2) (Siwei Lai, 2015).

The 20 news group dataset has a high features dimension and large number of classes to be categorized (20 distinct labels). A previous publication about enhanced SVM performance on text classification has reported the test accuracy they obtained for the Decision Tree, Random Forest, AdaBoost

models over the same dataset, which are 0.4069, 0.4527, 0.4033 respectively. They combined SVM with multiple strategies, and reported an average test accuracy of 0.67. In addition, they also offered results over prediction of 4 of the classes, with 2 classes highly correlated to each other and other 2 totally uncorrelated. Not surprisingly all models have received 10% of accuracy improvement, with their enhanced SVM maintaining highest accuracy. (Soumick Chatterjee, 2019)

As for IMDB dataset from Stanford, the reported accuracy of these models are significantly higher than those of 20 news group. Published literature reported a highest accuracy amongst RF, AB, DT of 0.8624(Md Fahimuzzman Sohan and Rahman1, 2018), and 0.8550 for SVM(Tarimer et al., 2019).

3 Experiments

3.1 Dataset

To compare the performance of the models, we performed text classification experiments using the 20 news group and IMDB dataset. Notice that both datasets already come with a training and a testing set, therefore we do not need to manually divide the datasets for training and testing.

- The *20 news group* dataset contains approximately 18 000 news documents evenly distributed across 20 news groups for both training and testing set.
- The *IMDB* dataset contains a total 50 000 movie reviews: 25 000 for each of positive (12500 train and test) and negative reviews (12500 train and test).

3.2 Feature Extraction and Dimension Reductions

A brief description on terminologies that would appear in this report.

- **Stop words** are the most common words appeared in a language. For English, stop words can be *a, and, the, of,* etc. These words usually do not contain significant semantics that influence outcomes of classification algorithms.
- **Lemmatization** means to remove inflectional endings (or sometimes starts), and attempt to recover the *lemma* in the spelling form recorded by dictionaries. Eg. *foxes* would become *fox* after lemmatization.
- **Stemming** means to heuristically chop off parts of the word and glue back some affixes to recover the *stem*. Eg. *relational* becomes *relate*, and *ponies* becomes *poni* after stemming.
- **Bag of words** refers to all words appear in the dataset. For each text data, we can transform it to a count or a binary (presence or absence) vector of bag-of-words.
- **Term frequency–inverse document frequency (TF-IDF)** refers to statistical importance of a word from a document in a dataset. Formula for this can be found online.(Kamran Kowsari and Brown, 2019)

To extract important features (words), we first removed the English stop words. Next, we applied three different strategies to extract the words of the dataset: 1) keep each text as it is which we refered as a normal prepossess, 2) to apply lemmatization and 3) to apply stemming to each word from a text. All the above tasks utilized the nltk python package (Loper and Bird, 2002).

After feature extraction, we used the *CountVectorizer* class from Sci-Kit Learn to transform the datasets into matrix form X (namely X_{train} and X_{test}). Each column j of the matrix refers to a word found in the dataset, each row i refers to a text document, and each entry X_{ij} indicates the occurrence of the word j appeared in document i. Notice that, for both 20 news group and IMDB, we only fitted *CountVectorizer* on the training set, then transformed both the training and test sets to get X_{train} and X_{test} respectively. Because of this, words that appeared in the testing set but not in the training set would not have corresponding columns in the matrix. To further reduce the sparsity and the dimension of the matrix, we chose to select words from the training set that appeared at least 20 times (a heuristic threshold). Doing so can help us save computational effort and alleviate the Curse of Dimensionality. We noticed that there are many better dimension reduction algorithms such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). However, for this project, we would like to focus on evaluating the performance of each classifiers, rather than addressing the issues of time complexity and memory consumption in which PCA and LDA aim to solve. In the end we used the *TfidfTransformer* class from Sci-Kit Learn to transform each entry of X into TF-IDF.

	DT	LR	SVM	AB	RF
Best hyperparameters (20 news group)	criterion: gini max depth: 100 FS: stemming	C: 0.5 penalty: l2 FS: stemming	C: 0.5 loss: squared hinge FS: stemming	learning rate: 0.5 n estimator: 100 FS: stemming	criterion: gini max depth: 100 n estimator: 100 FS: stemming
Train accuracy (20 news group)	0.71663	0.77320	0.93141	0.5	0.93327
Test accuracy (20 news group)	0.40321	0.60303	**0.63170**	0.43508	0.57793
Best hyperparameters (IMDB)	criterion: gini max depth: 25 FS: normal	C: 0.5 penalty: l2 FS: normal	C: 0.5 loss: squared hinge FS: normal	learning rate: 1 n estimator: 100 FS: normal	criterion: entropy max depth: 100 n estimator: 100 FS: lemmatizing
Train accuracy (IMDB)	0.88664	0.88860	0.94288	0.83040	0.99948
Test accuracy (IMDB)	0.72812	0.86788	**0.88144**	0.82472	0.84948

Table 1: Performance of the 5 classifiers in 20 news group and IMDB datas. The train accuracy and test accuracry are obtained under that best model output by *GridSearchCV* on the whole training and testing set respectively. The values in bold indicate the best test accuarcy for each dataset. FS stands for feature selection.

Again *TfidfTransformer* is fitted on the X_{train} first, then transforms both X_{train} and X_{test}. As a result, each word in the testing set would have the same statistical importance as it is in the training set.

3.3 Classifier Selection and Evaluations

For this project, we used the Sci-Kit Learn implementation for the five classifiers. In order to reproduce the result we get, we have fixed the random seed and random state to be 2020.

To select the best possible model for each dataset, we tuned the following hyperparameters for each classifier on the training set using the *GridSearchCV* from Sci-Kit Learn. In short, *GridSearchCV* would try all the combinations of parameters using 5-fold cross validation on the training set, and output the model that has the best mean score (accuracy) on the validation set.

- *DT*: criterion (Gini or entropy) and max depth (25, 50 or 100).
- *LR*: regularization (l1 or l2) and regularization strength ("C" parameter varying among 0.001, 0.005, 0.01, 0.05, 0.1, and 0.5).
- *SVM*: loss (hinge or squared hinge) and regularization strength ("C" parameter varying among 0.001, 0.005, 0.01, 0.05, 0.1, and 0.5).
- *AB*: number of estimator (25, 50 or 100) and learning rate (0.01, 0.05, 0.1, 0.5 or 1).
- *RF*: criterion (Gini or entropy), number of estimator (25,50 or 100) and max depth (25,50 or 100).

We chose these hyperparameters since they are expected to affect most on model performances. Though there are a lot of other hyperparameters that could be tuned, adding them to the grid search would have drastically increased the computational cost of the validation process. Therefore, those considered to be less effective were just set to default values as in Sci-Kit Learn implementation.

After tuning of the hyperparameters, we evaluated the accuracy of the best models on the whole training set and test set for both 20 news group and IMDB.

4 Results

All experimental results of post-tuning models on the 2 datasets are summarized in Table 1. Table 1 has also indicated best performing hyperparameters for each model under different datasets.

4.1 20 News Groups

Amongst the chosen models, SVM exhibited a winning accuracy at 0.6317 for 20 news group. During the tuning of hyperparameters, we noticed that a C value of 0.5 together with squared hinge loss function would produce best prediction results for SVM, regardless of our choice on feature selection methods. In addition, extracting features with stemming allowed all models to achieve their best

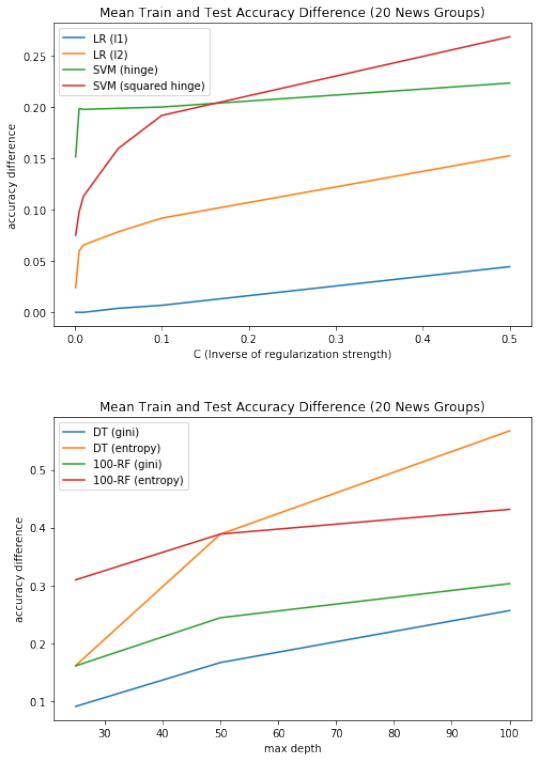

Figure 1: The difference between the mean train and val accuracy during CV over different regularization strength for different models. The results is obtained with normal preprocess technique.

performances in using 20 news group. Lastly, we observed that RF performed better than DT for both training and testing.

For DT, LR, SVM and RF, their train accuracy is inconsistent with their test accuracy. This indicates a problem of overfitting for this dataset (Table 1). This problem can also be shown by looking at the training and validation accuracy during hyperparameter tuning, figure 1. Typically DT, LR, SVM and RF are fit by minimizing a loss function averaged over the training data, therefore it is not surprising to see these models having an overfitting issue if the training set is not generalized enough. However, as we can see in the figure 1, regularization can alleviate overfitting problem, but still does not significantly improve the performance on the test set. On the other side, we saw that AB does not suffer from the overfitting problem, since AB minimizes an exponential loss function which is very sensitive to changes in the estimated class probabilities. (Trevor Hastie)

4.2 IMDB Reviews

In the case of IMDB, the best accuracy (0.88144) also came from SVM. Here, the best setting of hyperparameters is consistent with that in 20 news group, except for the best choice for feature extraction here is the normal method (without lemmatization or stemming). Again, we observed that RF performed better than DT in both training and testing.

Even though from the IMDB results we still saw overall higher training accuracy than test accuracy in all models, their performance gaps between training and testing are significantly smaller than in the previous dataset. This may be due to some words in testing set but not training set having a high impact to the classification. As we mentioned earlier, during the stage of feature extraction, these words would not be extracted since the extractor does not see them in the training set. Consequently, these words may cause the classifier to perform a little bit poorer on the testing set than the training set. Therefore, even with slight overfitting, we think the scale of the gaps between training and testing performances are generally acceptable. Furthermore the gap between the train and test accuracy for AB is still the smallest among all model which confirm the fact that it is not trying to minimize the average loss in the training dataset.

5 Discussion and Conclusion

In this project, we evaluated the performances of the logistic regression, support vector machine, AdaBoost (with stump estimator), Random Forest, and Decision Trees on binary and multi-class text classification using the IMDB and 20 news group datasets respectively. By following our experimental design, SVM achieved the best performances on both datasets. In particular, its prediction accuracy for 20 news group significantly outran those of the rest. As one may expect, Decision Tree has overall the worst performance, since it is a simple model that is unstable and could easily overfit when the data is not generalized enough. Moreover, the Random Forest with bagging performed better than Decision Tree, confirming on the capability of bagging in reducing variance of models and improving prediction accuracy as suggested by mathematical theories.

In our results, we also observed an obvious issue of overfitting in the 20 news group data from all the models except for AdaBoost. As this problem is not observed in the other dataset (IMDB), we believe this is due to insufficient samples for each of the 20 classes. Notice that in IMDB, there are

12 500 reviews for each class, while only about 400 documents are for each class in the 20 news group. The low number of samples per class can lead to insufficient generalization of learning multi-class labels. Furthermore, in text classification, the feature matrix is usually sparse since each document contains only a tiny portion of the vocabulary of the whole dataset. This means that there is no sufficient features per sample for a classifier to generalize the classification task. As a result, these two issues can cause classifiers to have trouble performing well in general in multi-class text classifications.

In the future, we would like to investigate other feature extraction methods. An example of limitations of bag of words in terms of representing the features could be that it does not capture the similarity found for each word in a text, and the respect order of each words in a sentence. This would give the models a hard time to capture the semantic. Word embedding is a feature learning technique that aims at solving these limitations. Many existing algorithms have already been developed for word embedding such as Word2Vec, GloVe, and FastText (Kamran Kowsari and Brown, 2019). It would be interesting to see if they can improve the performance of the 5 classifiers. Moreover, we would like to investigate better dimension reduction techniques such as PCA and LDA that we have mentioned before.

6 Statement of Contributions

Si Yi Li: Data Preprocessing, Experiments, report write up
Matteo Cacciola: Data preprocessing, Experiments, report write up
Agnes Liu: Report write up

References

Mojtaba Heidarysafa Sanjana Mendu Laura Barnes Kamran Kowsari, Kiana Jafari Meimandi and Donald Brown. 2019. Text classification algorithms: A survey. *Information 2019*, 10:150–218.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, page 63–70, USA. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Md Tahsir Ahmed Munna Shaikh Muhammad Allayear Md. Habibur Rahman1 Md Fahimuzzman Sohan, Sheikh Shah Mohammad Motiur Rahman and Md. Mushfiqur Rahman1. 2018. *Next Generation Computing Technologies on Computational Intelligence*, chapter NStackSenti: Evaluation of a Multilevel Approach for Detecting the Sentiment of Users. Communications in Computer and Information Science.

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.

Kang Liu Jun Zhao Siwei Lai, Liheng Xu. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2267–2273, Palo Alto, California, USA. Association for the Advancement of Artificial Intelligence.

Debabrata Datta Soumick Chatterjee, Pramod George Jose. 2019. Text classification using svm enhanced by multithreading and cuda. *International Journal of Modern Education and Computer Science(IJMECS)*, 11:11–23.

Chengliang Zhang Tao Li and Mitsunori Ogihara. 2014. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20:2429–2437.

İlhan Tarimer, Adil Çoban, and Arif Kocaman. 2019. Sentiment analysis on imdb movie comments and twitter data by machine learning and vector space techniques.

Jerome Friedman Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics.