Applied Machine Learning Mini Project 2 Report

Abstract

In this project we implemented five different machine learning models, Support Vector Machine, Logistic Regression, Decision Tree, Random Forest, AdaBoost and investigated their classification performances on two text datasets. K-fold cross validation is employed along with randomized searching for hyperparameters tuning. Our experimental results show that Support Vector Machine approach achieves better performance than any other classifiers on both datasets. Its test accuracy is 0.88684 and 0.69583 for IMDB and News dataset respectively.

1 Introduction

In this project, 5 classification models—Support Vector Machine, Logistic Regression, Decision Tree, AdaBoost and Random Forest are implemented with *scikit-klearn*[1] and compared on two distinct datasets for text classification task. In the following part, hyper-parameters and their ranges for these 5 classifiers in our project are shown in detail.

1.1 Logistic Regression

Ten different hyper-parameters for Logistic Regression are selected and tuned to improve the accuracy. First, we specify the penalty either *L1-norm* or *L2-norm* associated with randomized regularization strength between 10^{-3} and 10^{3}. Second, we randomly choose solvers from *newton-cg, lbfgs, liblinear, sag,* and *saga* associated with two hyperparameters, the randomized tolerance for stopping between 10^{-11} and 10^{-4}, and the max iteration allowed for convergence between 50 and 200. Third, the features are also being tested which if a constant should be added to the decision function. Additionally, choosing between dual and primal formulation and whether or not using the multi-class feature are also in the consideration. Lastly, we test on whether to reuse the solution of the previous call to fit for the next iteration.

1.2 Decision Tree

Several hyperparameters are defined to grow a decision tree. We choose the best split or the best random split at each node, and measure the quality of a split either by Gini impurity or by the information gain. When looking for the best split, we use different number of features to consider as functions of total feature number. We also define different minimum increases. A node will be split if this action will induce a decrease of the impurity greater than or equal to this value. We choose settings with varied maximum depth of the tree, minimum number of samples required to split an internal node, minimum number of samples required to be at a leaf node, and maximum number of leaf nodes.

[1] https://scikit-learn.org/

1.3 Support Vector Machine

Five different parameters have been tuned to improve the accuracy for Support Vector Machine. Similarly to Logistic Regression, we choose dual or primal formulation and specify the penalty norm associated with the regularization random strength between 10^{-3} and 10^{3} besides the randomized tolerance for stopping between 10^{-11} and 10^{-4}.

1.4 AdaBoost

AdaBoost classifier has four main hyperparameters. Start from the base estimator where the boosted ensemble is built on from Linear SVC, Logistic regression, Multinomial Naive Bayes and Decision tree at the validation stage. The second parameter is the maximum number of iterators at which boosting is terminated. We test on this number from 10 to 400. The third one is learning rate which is a float number between 0 and 1 that shrinks the contribution of each classifier by itself. The last one is the boosting algorithm which can be set as either 'SAMME'(discrete boosting algorithm) or 'SAMME.R'(real boosting algorithm). Discrete SAMME AdaBoost adapts based on errors in predicted class labels whereas real SAMME.R uses the predicted class probabilities. The SAMME.R algorithm typically converges faster than SAMME, achieving a lower test error with fewer boosting iterations.

1.5 Random Forest

Random Forest implement averaging algorithms based on randomized decision trees. Thus, it possesses similar hyperparameters as a Decision Tree, but with several additional ones specifying whether bootstrap samples are used when building trees, whether to use out-of-bag samples to estimate the generalization accuracy, whether to reuse the solution of the previous call to fit and add more estimators to the ensemble or just fit a whole new forest, and the number of samples to draw from the training dataset to train each base estimator, as well as the verbosity when fitting and predicting.

1.6 Related Work

Multiclass classification is a central problem in machine learning, as applications that require a discrimination among several classes are ubiquitous. In machine learning, these include handwritten character recognition, part-of-speech tagging , speech recognition and text categorization. (Rennie and Rifkin, 2001) compared Naive Bayes and Support Vector Machines on the task of multiclass text classification two well-known text data sets, one of which is the same as the one we used in this project(20 Newsgroups). The paper concludes that SVM consistently outperforms Naive Bayes on

the multiclass classification task, but the difference in performance varies by matrix and amount of training data used. They achieved a lowest error of 0.125 on 20 Newsgroups dataset using the SVM with a 63-column BCH matrix. The main objective of text classification is to train the classifier on the basis of predefined categories. The various application domain of text classification generally includes web page classification, spam e-mail filtering, author identification, topic detection, etc. One of the biggest challenge is this field is high dimensionality of feature space due to the large number of features. This will increases the complexity of machine learning methods used for text classification and reduces the accuracy due to redundant or irrelevant terms in the feature space. To solve this problem, feature extraction and feature selection methods should be well practiced (Shah and Patel, 2016).

The rest of this report is organized as follows: Section 2 gives brief descriptions of the datasets and how we preprocessed data and extract features, Section 3 introduces our experimental approaches. Section 4 reports experimental results, and finally in Section 5 we summarized our findings and provided future directions.

2 Datasets and Setup

2.1 Datasets Description

First dataset in this project is the 20 news group dataset, which is a collection of approximately 20,000 newsgroup documents, partitioned nearly evenly across 20 different newsgroups and split into a train subset and a test subset.

Another dataset IMDB, including movie review, was first introduced in (Maas et al., 2011). This dataset is used for binary sentiment classification, containing a subset of 25,000 highly polar movie reviews for training, and a subset of 25,000 reviews for testing.

2.2 Setup

We employed the bag-of-words model to extract features from text data. In this model, a text is represented as the numbers of occurrences of its words, disregarding grammar and even word order but keeping multiplicity. To build features, we first tokenized the strings to tokens. To achieve better results, we employed Porter Stemmer to cut affixes off the tokens and find their stems. We chose scikit-learn's *TfidfVectorizer* to vectorize and processed text data, which can re-weight the count features into floating point values suitable for usage by a classifier.

3 Proposed Approach

A pipeline composed of vectorizer, transformer, classifier was constructed. Randomized search cross validation was employed for hyper-parameter tuning. Apart from hyper-parameters for each classifier introduced in 1, we also tuned parameters for vectorizer including whether to delete stop-words, token frequency threshold, minimum length of tokens and n-gram range(only unigram or unigram plus bigram). We set the number of hyper-parameter set candidates as 200 for all classifiers. Apart from 5 required classifier, we also implemented Bagging model, whose base estimator is chosen from Naive Bayes, 5 required classifier with optimal hyper-parameters and the default Decision Tree model. After hyper-parameter tuning, the optimal parameters, which gives the highest mean validation accuracy, are selected to refit the model with the whole train data and then the refitted model is used to make predictions on test data.

4 Results

4.1 IMDB Dataset

The test accuracy for 5 classifiers plus Bagging model on IMDB Dataset is illustrated in Table 1. The associated training time is shown in Table 2 and optimal hyper-parameters in A. Support Vector Machine achieved the highest accuracy—0.88684.

4.2 News Dataset

The test accuracy for 5 classifiers plus Bagging model on News Dataset is illustrated in Table 1. The associated training time is shown in Table 2 and optimal hyper-parameters in A. Support Vector Machine achieved the highest accuracy—0.69583.

Table 1: Test accuracy of different classifiers on two datasets.

Classifier	**Dataset**	
	IMDB	News
LR	0.88680	0.68600
SVM	0.88684	0.69583
Decision Tree	0.75372	0.36272
AdaBoost	0.86000	0.60794
Random Forest	0.83528	0.54063
Bagging	0.85376	0.66994

4.3 Analysis and comparison

From Table 1, we can see that Support Vector Machine gives a better test accuracy for both datasets than other classifiers. Even though Decision Tree takes significantly less time to train, it gives the worst prediction result, which is 0.75372 for IMDB dataset and 0.36272 for News dataset. The reason for decision tree producing the worst result on both datasets is that decision tree is likely to overfit, by giving the large size of training dataset. Logistic Regression gives comparable prediction result to Support Vector Machine and takes less time to train. An unexpected result is that ensemble models including AdaBoost and Bagging cannot achieve

Table 2: Running Time in Seconds of different classifiers on two datasets for randomized search cross validation (200 hyperparameter candidates). The computational resource is desktop with Core i7-8700K processor for IMDB dataset and and laptop with 2.3GHz 8-core Intel Core i9 processor for News dataset.

Classifier	Dataset	
	IMDB	News
LR	1298.70	2197.43
SVM	1611.12	3689.71
Decision Tree	1028.19	567.92
AdaBoost	3456.41	14852.59
Random Forest	10196.02	3519.35
Bagging	9279.53	4918.78

better accuracy than Support Vector Machine and Logistic Regression, which may be due to the inconsiderable choice of base estimators.

5 Discussion and conclusion

In this work we designed and conducted a range of experiments to compare five traditional classifiers on two benchmark datasets. Experiment results consolidate our assumptions that Support Vector Machine takes considerable less time to train and achieves the highest accuracy on both datasets.

Our conclusions can be meaningful for choosing suitable model for binary and multi-class text classification task. Future directions could be focused on comparing Support Vector Machine with more complex classification models like Neural Network on text classification datasets.

6 Statement of contribution

Tianchi Ma and Qiutan Wu are responsible for dataset preparation and prepossessing. Everyone worked on implementing the five classifiers model and hyper-parameter tuning. All team members contributed to report writing.

References

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-volume 1*, pages 142–150. Association for Computational Linguistics.

Jason DM Rennie and Ryan Rifkin. 2001. Improving multiclass text classification with the support vector machine.

Foram P Shah and Vibha Patel. 2016. A review on feature selection and feature extraction for text classification. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 2264–2268. IEEE.

A Optimal hyperparameters for different models on the two datasets

Hyperparameter	Value
Classifier Hyperparameters	
base_estimator	None
bootstrap	False
bootstrap_features	True
max_features	0.1847899364371668
max_samples	0.7880331950127928
n_estimators	254
warm_start	True
Vectorizer Hyperparameters	
max_df	0.9
min_df	0.00022740427112902652
ngram_range	(1, 1)
stop_words	None
token_pattern	'\\w{2,}'

(a) IMDB

Hyperparameter	Value
Classifier Hyperparameters	
base_estimator	MultinomialNB
bootstrap	False
bootstrap_features	False
max_features	0.9120442729722144
max_samples	0.7892022585496922
n_estimators	209
warm_start	False
Vectorizer Hyperparameters	
max_df	0.6333333333333333
min_df	0.00011618768297197917
ngram_range	(1, 2)
stop_words	'english'
token_pattern	'\\w{2,}'

(b) 20 Newsgroups

Table 3: Optimal hyperparameters for Bagging.

Hyperparameter	Value
Classifier Hyperparameters	
C	3.420814845719859
dual	False
fit_intercept	False
l1_ratio	0.7919834177347317
max_iter	83
multi_class	'auto'
penalty	'l2'
solver	'lbfgs'
tol	1.494527510423322e-05
warm_start	True
Vectorizer Hyperparameters	
max_df	0.9
min_df	0.0007660870622914968
ngram_range	(1, 2)
stop_words	None
token_pattern	'\\w{2,}'

(a) IMDB

Hyperparameter	Value
Classifier Hyperparameters	
C	3.420814845719859
dual	False
fit_intercept	False
l1_ratio	0.5790356826861355
max_iter	57
multi_class	'auto'
penalty	'none'
solver	'newton-cg'
tol	1.2982284932575753e-11
warm_start	False
Vectorizer Hyperparameters	
max_df	0.6333333333333333
min_df	0.00010489404472121024
ngram_range	(1, 2)
stop_words	None
token_pattern	'\\w{2,}'

(b) 20 Newsgroups

Table 4: Optimal hyperparameters for Logistic Regression.

Hyperparameter	Value
Classifier Hyperparameters	
C	0.219350884329188
dual	False
fit_intercept	True
loss	'squared_hinge'
tol	2.65363295575828e-11
Vectorizer Hyperparameters	
max_df	0.6333333333333333
min_df	0.001260437855257886
ngram_range	(1, 2)
stop_words	None
token_pattern	'\\w{1,}'

(a) IMDB

Hyperparameter	Value
Classifier Hyperparameters	
C	1.0437490839744141
dual	True
fit_intercept	True
loss	'squared_hinge'
tol	1.5305045599216338e-05
Vectorizer Hyperparameters	
max_df	0.8111111111111111
min_df	0.0001839752552740118
ngram_range	(1, 2)
stop_words	None
token_pattern	'\\w{1,}'

(b) 20 Newsgroups

Table 5: Optimal hyperparameters for SVM.

Hyperparameter	Value
Classifier Hyperparameters	
criterion	'gini'
max_depth	44
max_features	0.89
max_leaf_nodes	100
min_impurity_decrease	1e-05
min_samples_leaf	25
min_samples_split	0.00010101694850126544
splitter	'best'
Vectorizer Hyperparameters	
max_df	0.9
min_df	0.002481847462706278
ngram_range	(1, 2)
stop_words	'english'
token_pattern	'\\w{2,}'

(a) IMDB

Hyperparameter	Value
Classifier Hyperparameters	
criterion	'gini'
max_depth	45
max_features	0.45
max_leaf_nodes	316
min_impurity_decrease	3.1622776601683795e-10
min_samples_leaf	45
min_samples_split	0.023962817716297567
splitter	'best'
Vectorizer Hyperparameters	
max_df	0.8555555555555556
min_df	0.0004650471697148696
ngram_range	(1, 1)
stop_words	None
token_pattern	'\\w{2,}'

(b) 20 Newsgroups

Table 6: Optimal hyperparameters for Decision Tree.

Hyperparameter	Value
Classifier Hyperparameters	
bootstrap	True
class_weight	None
criterion	'gini'
max_depth	2
max_features	'auto'
max_leaf_nodes	316
max_samples	0.6627008736438607
min_impurity_decrease	1e-09
min_samples_leaf	11
min_samples_split	0.003934472581244525
min_weight_fraction_leaf	0.0012404894684496458
n_estimators	158
oob_score	False
verbose	False
warm_start	False
Vectorizer Hyperparameters	
max_df	0.8111111111111111
min_df	0.00025784568863467064
ngram_range	(1, 1)
stop_words	None
token_pattern	'\\w{1,}'

(a) IMDB

Hyperparameter	Value
Classifier Hyperparameters	
bootstrap	False
class_weight	'balanced_subsample'
criterion	'gini'
max_depth	46
max_features	'sqrt'
max_leaf_nodes	10
max_samples	0.7488406611631547
min_impurity_decrease	3.1622776601683795e-10
min_samples_leaf	21
min_samples_split	0.0740293055500459
min_weight_fraction_leaf	0.0006942969584767
n_estimators	125
oob_score	False
verbose	True
warm_start	False
Vectorizer Hyperparameters	
max_df	0.7222222222222222
min_df	0.00017810062892972548
ngram_range	(1, 1)
stop_words	'english'
token_pattern	'\\w{1,}'

(b) 20 Newsgroups

Table 7: Optimal hyperparameters for AdaBoost.

Hyperparameter	Value
Classifier Hyperparameters	
bootstrap	True
class_weight	None
criterion	'gini'
max_depth	2
max_features	'auto'
max_leaf_nodes	316
max_samples	0.6627008736438607
min_impurity_decrease	1e-09
min_samples_leaf	11
min_samples_split	0.003934472581244525
min_weight_fraction_leaf	0.0012404894684496458
n_estimators	158
oob_score	False
verbose	False
warm_start	False
Vectorizer Hyperparameters	
max_df	0.8111111111111111
min_df	0.00025784568863467064
ngram_range	(1, 1)
stop_words	None
token_pattern	'\\w{1,}'

(a) IMDB

Hyperparameter	Value
Classifier Hyperparameters	
bootstrap	False
class_weight	'balanced_subsample'
criterion	'gini'
max_depth	46
max_features	'sqrt'
max_leaf_nodes	10
max_samples	0.7488406611631547
min_impurity_decrease	3.1622776601683795e-10
min_samples_leaf	21
min_samples_split	0.0740293055500459
min_weight_fraction_leaf	0.0006942969584767
n_estimators	125
oob_score	False
verbose	True
warm_start	False
Vectorizer Hyperparameters	
max_df	0.7222222222222222
min_df	0.00017810062892972548
ngram_range	(1, 1)
stop_words	'english'
token_pattern	'\\w{1,}'

(b) 20 Newsgroups

Table 8: Optimal hyperparameters for Random Forest.