Empirical Comparison of Supervised Learning Algorithms

[CS269 Spring 2013 Project1 Report]

Chenguang Shen
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
cgshen@cs.ucla.edu

1. SUMMARY

In this project I compare the performance of three supervised machine learning algorithms (random forest, SVM, and boosted tree) on three binary classification problems. The data sets are retrieved from the public UCI machine learning repository [1]. The experimental setting generally follows the work by Caruana et al. [2], and result is consistent with their observation with minor differences.

In this report I will introduce the detailed setting of all my experiments, and provide comparison with the result in [2]. For the difference in results, I will try to explain the possible reasons.

2. METHODOLOGY

2.1 Data Sets

I have chosen 3 data sets from the UCI data set [1], including ADULT, COV_TYPE and LETTER. The basic information of these 3 data sets are shown in Table 1.

The original ADULT data set on the UCI repository has 14 features, and is a binary classification problem. The features describes basic information of a person, and the label shows whether his/her annual earning is higher than 50K. However, 8 of the 14 features are multi-class categorical strings, therefore it is not optimized for some of the classifiers. In order to get better performance, I used an optimized version of the ADULT data set by Platt et al. [5], where continuous features are discretized into quantiles, and each quantile is represented by a binary feature. Also, a categorical feature with m categories is converted to m binary features. There revised data set has 123 binary features with value {0, 1}.

The COV_TYPE data set is a multi-class classification problem. It is used to predict forest cover type from its properites. It has been converted to a binary problem by treating the largest class as the positive and the rest as negative. I addition, all 14 features are scaled to [0..1] using the LIBSVM [3] scaling tool.

The LETTER data set is also multi-class classification problem. The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. In order to convert it to a binary classification problem, LETTER.p1 treats ÓOÓ as positive and the remaining 25 letters as negative, yielding a very unbalanced problem. LETTER.p2 uses letters A-M as positives and the rest as negatives, yielding a well balanced problem. The data in both case are not scaled.

PROBLEM	# ATTR	TRAIN SIZE	TEST SIZE
ADULT	123	5000	25000
COV_TYPE	54	5000	25000
LETTER.P1	16	5000	14000
LETTER.P2	16	5000	14000

Table 1: Summary of Data Sets

Since the LETTER data set is evolved into LETTER.p1 and LETTER.p2, there are actually four problems in total. For each data set, I choose the training and testing data as a whole form the data set file, and randomly select 5000 samples as training set. The random selection process is done by Python's *random.shuffle()* method.

2.2 Learning Algorithms

I implement all three algorithms in Python, with the help of scikit-learn package [4]. The parameter setting of each of the classifiers follows [2], and is discussed below:

Random Forest
The forests used in this project have 1024 trees. The size of the feature set considered at each split is 1, 2, 4, 6, 8, 12, 16 or 20. When this size exceeds the total number of features in a specific data set, I just use the maximum number of features

SVM
I use the following kernels: linear, polynomial degree 2 & 3, radial with width {0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2}.

Boosted Tree
I used scikit-learn's *GradientBoostingClassifier* implementation for boosted trees. In each stage a binary regression trees are fit on the negative gradient of the binomial deviance loss function. Boosting can overfit, so I consider boosted trees after 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 and 2048 steps of boosting.

Note that I am not using any calibration method for these classifiers.

2.3 Performance Metrics

I use ACC, FSC, ROC, and APR as performance metrics for the three algorithms:

ACC
Accuracy of the classification.

FSC
F-score of the classification (treating the binary classification problem as a true-false problem). I use F1 score in this project.

ROC
Area under the ROC (Receiver Operating Characteristic) curve, treating the binary classification problem as a true-false problem.

APR
Average precision of the classification, treating the binary classification problem as a true-false problem.

3. PERFORMANCE BY METRIC

To test the performance of each algorithm, I used the same trial method as in [2]. For each algorithm, I conduct a 5-fold cross validation and obtain 5 trials. That is, for the 4000 training samples and 1000 testing samples in each cross validation, I will try to train the classifier with all possible parameters, and use the 1000 testing samples to obtain the best classifier. The metric for selecting best classifier is just aver score over the 4 metrics, and no calibration method is used. Then the best trained classifier is used to test the large testing set.

In this process, I get the mean metric score for each classifier over all 3 problems, both for the cross-validation, and for the larger testing set. These two kinds of score are listed separately.

3.1 Cross-validation scores

Table 2 shows the 5-fold CV scores for each learning algorithm by metric, which is the average over four problems. This is the score of applying one classifier with its best parameter setting on the small test set (1000 samples) in each 5-fold cross-validation. Random forest has highest scores on every metric, and is ranked as the best algorithm among these three. Boosted tree scores better than SVM on accuracy and precision, but is worse on area under ROC curve and f-score. On average boosted tree scores a little better than SVM.

3.2 Test scores

Table 3 shows the test scores for each learning algorithm by metric, which is the average over four problems. This is the average score of using the best classifier trained from 5-fold cross-validation to test on the large test data set. In Table 3, random forest has the best accuracy and precision, while SVM has the best are under ROC curve, and f-score. On average random forest is the best among these three, while SVM and boosted tree are ranked in 2nd and 3rd place, respectively. Note that the score difference here between these three algorithms are pretty small.

3.3 Analysis

In paper [2], similar result on testing set is presented, which is summarized in Table 4. Note that because I am not using any calibration method here, I have chosen the corresponding result from [2] where no calibration is used. The result in Table 4 should be compared with the Table 3 here.

In both Table 3 and Table 4, random forest is ranked as the best algorithm. However, the rank of SVM and boosted tree is reversed in my result (Table 3). This is probably because I am using scikit-learn's *GradientBoostingClassifier* implementation, where the type of tree is not clear. However, in [2] the author consider more difference kinds of trees, such as BAYES, ID3, CART, CART0, C4, MML, and SMML. In each 5-fold cross-validation stage the best tree will be selected out of these types. Therefore it is reasonable to assume some other types of decision tree will have better performance than the tree used by *GradientBoostingClassifier*. In addition, in Table 4 boosted tree has best scores in APR and FSC, but not in Table 3. This is probably because of the same reason stated above.

In general, the MEAN numbers in Table 3 are smaller than those in Table 4. This is correct because I am not normalizing the scores in my result. However, there is some exceptions, such as the accuracy for all three classifiers. One reason is that I am not using exactly the same data set as in [2]. For ADULT, I am using the optimized data set instead of the original one. For COV_TYPE, I am using the scaled version of the original dataset. These two optimized version of data sets have resulted in the improved performance in accuracy of my classifier, especially for SVM, which is known to perform poorly with non-scaled data set.

4. PERFORMANCE BY PROBLEM

Table 5 and Table 6 show the score for each algorithm on each of the 3 test problems in 5-fold validation and test, respectively.

4.1 Cross-validation scores

Table 5 shows the 5-fold cross-validation scores for each learning algorithm by problem, which is the average over four metrics. For COV_TYPE and LETTER.p1, random forest has the best average performance. For ADULT and LETTER.p2, SVM has the best average performance.

4.2 Test scores

Table 6 shows the test scores for each learning algorithm by metric, which is the average over four problems. For COV_TYPE, random forest has the best average performance. For ADULT, LETTER.p1, and LETTER.p2, SVM has the best average performance.

4.3 Analysis

The corresponding result on performance-by-problem for testing set in [2] is shown in Table 7. Note that because they are using more metrics (in total 8 metrics) than I do, it is not particularly meaningful to compare the result by each problem. However, we can still find some clue about the difference.

The most important observation is that SVM is performing better than boosted tree in my experiment using the big testing set in Table 6, while in Table 7 SVM performs much worse than boosted tree. Once again I concludes the

MODEL	ACC	ROC	APR	FSC	MEAN
RF	0.908	0.871	0.875	0.839	0.873
BST-DT	0.901	0.862	0.866	0.830	0.865
SVM	0.896	0.868	0.857	0.834	0.864

Table 2: 5-fold CV scores for each learning algorithm by metric (average over four problems)

MODEL	ACC	ROC	APR	FSC	MEAN
RF	0.890	0.836	0.844	0.794	0.841
SVM	0.884	0.843	0.833	0.801	0.840
BST-DT	0.885	0.835	0.839	0.794	0.838

Table 3: Testing scores for each learning algorithm by metric (average over four problems)

MODEL	1ST	2ND	3RD
RF	0.392	0.200	0.408
SVM	0.383	0.283	0.333
BST-DT	0.225	0.517	0.259

Table 8: Bootstrap analysis of overall rank by mean performance across problems and metrics

reason for this is because I am using some optimized data set, which could improve the performance of SVM significantly. SVM generally does not perform well when data is not scaled. By using the optimized ADULT data set and the scaled COV_TYPE data set, the performance of SVM is significantly improved.

5. BOOTSTRAP ANALYSIS

Similar to the [2] paper, I also finish the bootstrap analysis. Since I only have 4 problems, I did not do sampling with replacement on the problem set. Instead, I run each classification algorithm 30 times with each of the 4 problems, using sampling with replacement to get the input data set. For each of the run, I also use 5-fold cross-validation to obtain 5 trials, and use the best trained classifier to test on the large test set. I only run each algorithm 30 times on each problem because it is pretty expensive to increase the running times.

The result of the bootstrap analysis is shown in Table 8. The rank of the 3 algorithms (RF, SVM, BST-DT) is consistent with the result in Table 3 and Table 6, showing that the results in the previous performance-by-metric and performance-by-problem experiments are stable. Since the paper [2] is using the best trained classifier for each algorithm, it is not appropriate to compare it with the result here, since I am not using all parameters and there is not calibration in my algorithm.

6. CONCLUSION

Generally, the results in this project are consistent with the result shown by Caruana et al. [2]. Without any calibration, random forest always has the best overall performance. In addition, random forest has the simplest algorithm of all these three ones. However, it also takes the most time to train.

The inconsistency of my result compared to the Caruana paper comes from three aspects:

1. Due to the time limit, I am not using all parameters described in the Caruana paper, and resulting bad performance of boosted tree.
2. I am not doing calibration for all algorithms.
3. I am using optimized and scaled data sets for ADULT and COV_TYPE, resulting improved performance of SVM.

7. REFERENCES

[1] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[2] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 161–168, New York, NY, USA, 2006. ACM.

[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[4] 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. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] J. C. Platt. Advances in kernel methods. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in kernel methods*, chapter Fast training of support vector machines using sequential minimal optimization, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

MODEL	ACC	ROC	APR	FSC	MEAN
RF	0.872	0.951	0.931	0.934	0.922
BST-DT	0.834	0.963	0.938	0.816	0.888
SVM	0.817	0.938	0.899	0.804	0.864

Table 4: Normalized performance by metric result in [2]

MODEL	COV_TYPE	ADULT	LETTER.p1	LETTER.p2	MEAN
RF	0.828	0.736	0.972	0.958	0.874
BST-DT	0.800	0.740	0.961	0.957	0.864
SVM	0.774	0.746	0.972	0.963	0.863

Table 5: 5-fold CV scores of each learning algorithm by problem (averaged over for metrics)

MODEL	COV_TYPE	ADULT	LETTER.p1	LETTER.p2	MEAN
RF	0.818	0.713	0.931	0.902	0.841
SVM	0.772	0.742	0.933	0.912	0.839
BST-DT	0.794	0.733	0.925	0.901	0.838

Table 6: Testing scores of each learning algorithm by problem (averaged over for metrics)

MODEL	COV_TYPE	ADULT	LETTER.p1	LETTER.p2	MEAN
RF	0.876	0.946	0.883	0.922	0.907
BST-DT	0.874	0.842	0.875	0.913	0.876
SVM	0.696	0.819	0.731	0.860	0.776

Table 7: Normalized performance by problem result in [2]