An Exploratory Study of Supervised Machine Learning Algorithms

Reznikov, Daniel
drezniko1@ucsd.edu
Yang, Yunfan
yuy130@ucsd.edu

UCSD COGS 118A
WINTER 2016
Professor Tu

Abstract

There are many powerful applications of machine learning models and techniques in today's data-driven, digital world. As machine learning engineers, we have to understand the efficacy and methodology of the practical applications of machine learning algorithms. We present an empirical verification of the 20 of the almost 100 results published by Caruana and Niculescu-Mizil in their 2006 paper, "An Empirical Comparison of Supervised Learning Algorithms." We programatically reproduce their experimental results, comparing our results in a space of several important model scores and analyzing our methodology and training of model hyper-parameters.

1 Introduction

While Caruana and Niculescu-Mizil implemented a broad, wide-reaching and comprehensive empirical survey of the performance of nearly 2-dozen machine learning techniques against 11 distinct datasets, the scope of our project is more limited. We aim to reproduce a subset of their results by training a handful of commmonplace, well regarded models against 4 datasets, measuring each experiment across 5 statistical metrics.

Our aim is to understand the implementation pipeline for data science including data-cleansing, application of third party libraries, hyper-parameter tuning, and analysis of results. A secondary goal is to reproduce the results in Cauruana and Niculescu-Mizil's paper and, where our results differ, dive into our methodology in order to extrapolate the consequences of our decisions and how they could explain differences in learning rates.

2 Methodology

2.1 Technology

Each experiment was written in a Jupyter Notebook and executed on Amazon's EC2 Cloud Computing service, where we rented a server with 16 cores and 32GB of RAM. This hardware choice was important because we found that some classifiers were pain-stakenly slow on our local machines and accessing this resource significantly expedited our iteration cycles.

2.2 Learning Algorithms

Using well-known Python libraries, we implemented a half-dozen classification models. We implemented:

BDT - Boosted Decision Tree:

We use the AdaBoostClassifier provided in scikit-learn and apply gridsearch cross-validation to select the hyper-parameter, steps of boosting from the range 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, and 2048.

KNN - K Nearest Neighbors:

We use KNeighborsClassifier provided in scikit-learn and apply gridsearch cross-validation to choose odd value of k from 1 to 30. We use the Euclidean distance between each test point to all of its training neighbors in our computations.

NN - Neural Network

We use MLPClassifier provided in the scikit-learn 0.18 dev version and apply gridsearch cross-validation to choose the learning rate. We applied a one hidden layer ANN with 640 neuron with Stochastic gradient descent algorithm and early stopping option.

RF - Random Forest Classifier

We use RandomForestClassifier provided in scikit-learn and apply gridsearch cross-validation to choose the hyper-parameter, the size of feature set considered at each split, ranging from 1,2,4,6,8,12,16,20. The forest has 100 leaves.

SVM - Support Vector Machine

We use linearSVC provided in scikit-learn and apply gridsearch cross-validation to choose the hyper-parameter, C, from the range 0.1, 1, 2, 3, 4, 5, 6, 7. We also compared linearSVC with SVM with linear kernel, and we decided to use linearSVC in our study because SVM with linear kernel uses a kernelized implementation that is too expensive to run on large datasets.

XGBoost

We use XGBoost's scikit-learn interface and gridsearch cross-validation to choose hyper-parameter, steps of boosting from the range 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048. XGBoost is a newly emerged paralleled boosting algorithm providing significant speedup comparing to AdaBoost.

2.3 Performance Metrics

We ran each of 6 classifiers against each of 5 datasets or, equivalently, we ran 30 experiments. In each experiment, we used 5-fold cross-validation to tune the relevant hyper-parameters. Once trained, we ran each model against the withheld test dataset and computed 5 statistical measures of performance. Those measures are:

1. Accuracy - The true positive classification rate on the test dataset.
2. F1 Score - Measures the test's accuracy.
3. Precision - Measures the fraction of retrieved instances that are relevant.
4. Recall - Measures the fraction of relevant instances that are retrieved (sensitivity).
5. ROC - The area under the ROC Curve is another measurement of accuracy.

2.4 Data Sets

We ran our experiments against 5 datasets from the UC-Irvine Machine Learning Repository. Three of these are the exact datasets Caruana and Niculescu-Mizil used, allowing for direct and unambiguous comparison of results while controlling for noise signals in the underlying dataset. 'ADULT' is a dataset containing 15 demographic metrics as features, and a label distinguishing US adults based on whether or not their annual income exceed $50,000 USD.

In the 'COV_TYPE' dataset, the problem set changes, providing labels corresponding to various forest cover types. We are given 58,000 data points each with 54 quantitative wilderness data features, and one of 7

distinct labels corresponding to 7 different forest cover type classifications. In our experiments, we changed the multi-class problem to a binary classification problem by distinguishing between the most popular forest cover type (Lodgepole Pine) versus the rest. We also limited the dataset from the given 58,000 to 30,000 random data points to enable reasonable computation times.

3 Experiment Analysis

Table 1: Experiment Data

Model	Data Set	Accuracy	F1-Score	Precision	Recall	ROC Area	Train T	Pred T
BDT	Adult	0.852	0.847	0.846	0.852	0.770	14.254	0.228
BDT	Breast Cancer	0.950	0.950	0.950	0.950	0.947	2.912	0.001
BDT	COV_TYPE	0.861	0.862	0.863	0.861	0.862	41.376	0.497
BDT	Letter P1	0.983	0.982	0.982	0.983	0.860	31.919	0.129
BDT	Letter P2	0.821	0.821	0.821	0.821	0.821	38.984	0.444
KNN	Adult	0.778	0.710	0.803	0.778	0.563	1.276	1.204
KNN	Breast Cancer	0.630	0.487	0.397	0.630	0.500	0.211	0.001
KNN	COV_TYPE	0.912	0.912	0.912	0.912	0.911	1.093	0.510
KNN	Letter P1	0.990	0.990	0.990	0.990	0.931	1.526	1.060
KNN	Letter P2	0.959	0.959	0.959	0.959	0.959	0.955	0.766
NN	Adult	0.751	0.646	0.808	0.751	0.504	48.455	2.034
NN	Breast Cancer	0.630	0.487	0.397	0.630	0.500	0.225	0.005
NN	COV_TYPE	0.832	0.832	0.834	0.832	0.833	31.385	1.135
NN	Letter P1	0.962	0.944	0.926	0.962	0.500	8.973	0.224
NN	Letter P2	0.839	0.839	0.839	0.839	0.839	26.175	0.225
RF	Adult	0.830	0.825	0.823	0.830	0.748	2.837	0.305
RF	Breast Cancer	0.965	0.965	0.965	0.965	0.963	0.481	0.007
RF	COV_TYPE	0.922	0.922	0.922	0.922	0.921	3.709	0.358
RF	Letter P1	0.985	0.984	0.985	0.985	0.818	1.743	0.121
RF	Letter P2	0.943	0.943	0.943	0.943	0.943	2.379	0.188
SVM	Adult	0.748	0.641	0.560	0.748	0.500	0.475	0.006
SVM	Breast Cancer	0.369	0.119	0.136	0.369	0.500	0.214	0.001
SVM	COV_TYPE	0.525	0.362	0.276	0.525	0.500	0.664	0.087
SVM	Letter P1	0.962	0.944	0.926	0.962	0.500	0.417	0.016
SVM	Letter P2	0.503	0.336	0.253	0.503	0.500	0.418	0.015
XGBoost	Adult	0.858	0.852	0.852	0.858	0.775	19.809	0.043
XGBoost	Breast Cancer	0.959	0.959	0.959	0.959	0.954	1.241	0.001
XGBoost	COV_TYPE	0.916	0.916	0.916	0.916	0.915	16.592	0.179
XGBoost	Letter P1	0.991	0.990	0.990	0.991	0.909	6.627	0.091
XGBoost	Letter P2	0.935	0.934	0.935	0.935	0.935	6.772	0.097

3.1 Empirical Comparison

Here we tabulate the mean of the testing accuracy of each model, aggregated across all of the testing data sets. Adjacent to our results are the empirical results from the Caruana and Niculescu-Mizil paper.

Table 2: Accuracy Per Learning Algorithm

Model	Accuracy Reznikov/Yang	Accuracy Caruana/Mizil
BDT	0.893	0.843
KNN	0.853	0.756
NN	0.802	0.803
RF	0.929	0.861
SVM	0.621	0.817
XGBoost	0.931	N/A

We report excellent results overall, beating their results outright in 3 out of 5 comparable models. Our best performing model was XGBoost with a 93.1% accuracy but this algorithm is relatively new so it was not tested by Caruana and Niculescu-Mizil. The largest relative margin was for the Random Forest model. Here, our testing accuracy was 92.9% compared with their 86.1%, beating their results by 9.7%.

The only model that performed worse was Linear SVM. Our average accuracy was 62.1%, almost 20% worse than the results by Caruana and Niculescu-Mizil. The cause is not clear given that we trained the 'C' hyper-parameter with 5-fold cross-validation, the same as in the other models. Our first experiments used scikit-learn's svm library but we were hard-pressed when the dataset would not finish computing after hours of run-time. We swapped this library out in exchange for the a simpler method in the same library called 'linearSVC'. In the first method, the model was using a linear kernel mapping which took significant computation resources. In 'linearSVC' we are actually still doing linear fitting, but without employing the kernel mapping tick. This model runs significantly faster $\big(O(n^3)$ compared with $O(n)\big)$ but it still does not produce accurate results. Our final conclusion is that sklearn's svm module is not an effective implementation of Support Vector Machines.

In analyzing our Training and Prediction times for each experiment we found several trends. First, XGBoost runs faster than Boosted Decision Tree, which makes sense because the latter is implemented in parallel, training each tree simultaneously instead of serially. For Neural Networks, we observe that while it takes a significant amount of time to train, its relative classification time is actually quite fast. A similar training to prediction ratio for K Nearest-Neighbors shows us that this model is fast to learn but slow to classify. Finally, we note that while linear SVM is very fast to both train and test, its accuracy is, as previously noted, the worst in our experiment set.

Acknowledgments

We want to expressly acknowledge the teaching and instruction provided by Prof. Zhuowen Tu. Additionally we want to highlight our exceptional TA, Daniel Maryanovsky whose mentoring and patience was instrumental for our success throughout this project and the course at large.

4 Conclusion

In an evolving field, machine learning techniques are developing rapidly. Their ease of implementation is remarkable. With the wide-spread use and adoption of open source libraries - robust, efficient, well-documented, and effective implementations of a plethora of algorithms are available to the mass public. Superimposed with the massive growth of computational resources available to us through platforms like Amazon's EC2 (which was used in this study), this field has come tremendously far since the time of the Caruana - Niculescu-Mizel article in 2006.

We have found that many historically good learning techniques can be significantly improved through cross-validation. Libraries like Python Scikit-Learn enable a software engineer to deploy robust learning algorithms against commonly available data sets.

Our best performing model was XGBoost, but the most impressive model was Random Forest which can be tuned to excellent accuracy without significant delay in training or classification times.

References

[1] Caruana, R., Niculescu-Mizil A. (2006). "An Empirical Comparison of Supervised Learning Algorithms" ICML 2006: 161-168.

[2] Caruana, R., Karampatziakis, N., Yessenalina, A. (2008). "An Empirical Evaluation of Supervised Learning in High Dimensions" ICML 2006: 161-168.