Andrew Washington

COGS 118A

Professor Tu

June 17, 2017

An Empirical Comparison of Supervised Learning Algorithms on a Laptop Computer

Abstract:

Machine learning researchers use large amounts of computational power to build powerful models. Machine learning engineers, although, do not always have large computational resources at hand. Other times it is simply excessive to use expensive machines, when a comparable model can be built on a basic laptop. This paper is meant to demonstrate that decent models can be built for little cost, regarding both time and money.

Introduction:

Many machine learning engineers and researchers take advantage of large computational resources to run extremely computationally intensive machine learning algorithms (eg. deep learning). On the other hand, there has also been research into running extremely computationally efficient algorithms on small embedded processors. Little work, although, has been done in the middle ground: machine learning algorithms that can provide decent results on a typical laptop computer or modern workstation.

This goal of this experiment is to show that decently accurate models can be made and trained quickly without large amounts of computational resources. Of course, if one requires a great amount of accuracy, he or she can spend a great deal of time and money developing better computational tools, both physical and algorithmic. But this paper is to help demonstrate that decent accuracy can be achieved with minimal effort using open-source tools at little cost.

This experiment's design is meant to simulate a typical Data Scientist's workflow on a standard workstation. Because not all Data Scientists are experts in the field of Machine Learning, all models built will be built using modern open-source libraries. The SciKit-Learn library was chosen because of its widespread use in the field of Data Science as well as its simplicity and modularity.

Methods:

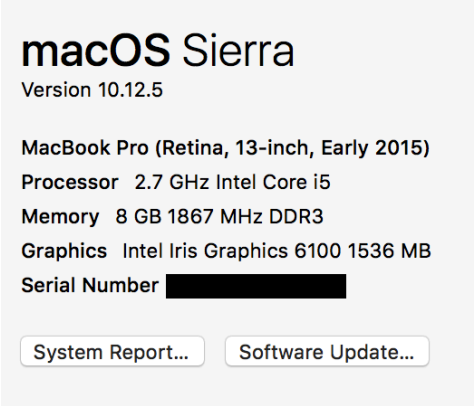
The following algorithms were tested: K-Nearest Neighbors (KNN), Random Forests, and AdaBoost. All algorithms were from the SciKit-Learn library version 0.18.1 (available at http://scikit-learn.org/stable/). The KNN models tested included 1, 3, 5, and 7 neighbors. Each number of neighbors was tested with three search algorithms (KD Tree, Ball Tree, and brute force) to observe effects on timing. The parameter focused on in the Random Forest models is the maximum depth of the tree. The tested values range from one to one hundred with a step size of one. The parameter focused on in the AdaBoost models is the number of estimators the algorithm could use. This parameter is tested in the range from one to three hundred with a step size of one.

The three datasets used in this experiment are: Adult, Ionosphere, and Arrhythmia. Each of the datasets used are from the UC Irvine repository (available at https://archive.ics.uci.edu/ml/datasets.html). For each dataset, each model was trained on 80% of the full dataset (randomly sampled) and tested on the remaining 20% of the dataset. The Adult dataset was encoded using a one-hot encoding scheme and, after encoding, had 41 dimensions and 32,561 observations. The Ionosphere dataset comes in a normalized, numerical form and has 34 dimensions and 351 observations. The Arrhythmia dataset has 279 dimensions and 452 observations. These three datasets were chosen so that one would have many dimensions

(Arrhythmia), one would have many observations (Adult), and one would have a smaller number of both (Ionosphere).

Each model was first tuned using cross-validation on the Adult dataset to get around the same accuracy as demonstrated in Caruana and Niculescu-Mizil's experiment. Next, the models were tuned on the remaining two datasets. Then each model was re-trained and tested using the optimal hyper-parameters while being timed. The accuracy and timing reported are those of the models with the highest cross-validation accuracy. The exception is for the KNN models, which were run and timed with all hyper-parameter and dataset combinations.

All tests were done using Python 3.6.0 in a Jupyter Notebook (Jupyter version 4.4.0, server running on Python 2.7.13) on macOS Sierra version 10.12.5. The computer used for testing is a MacBook Pro Retina Display 13" Early 2015-Version with a 2.7 GHz Intel i5 processor and 8GB DDR3 RAM. This laptop was chosen because it has specifications that are typical for a modern workstation used by a student, researcher, or worker in industry. Other libraries used include: Pandas (version 0.19.2), NumPy (version 1.12.0), and SciPy (version 0.18.1). The most important of these libraries (to this experiment) is NumPy, which was used to store the data. Pandas and SciPy were only used to load the data from .csv and .mat files.



Results:

Dataset:	Adult			Ionosphere			Arrhythmia		
Metric:	Accuracy (%)	Training Time (ms)	Prediction Time (s, N=6512)	Accuracy (%)	Training Time (ms)	Prediction Time (ms, N=70)	Accuracy (%)	Training Time (ms)	Prediction Time (s, N=90)
KNN	78.0	**20.6**	**3.58**	**97.1**	**1.12**	36.3	**80**	**4.32**	**0.0636**
RF	86.2	281.0	7.18	**97.1**	1.75	90.	**80**	25.7	0.114
Ada	**86.9**	4410.	38.5	94.3	28.3	69.	75.6	28.3	0.426

Table 1: Speed and Accuracy Results, All Algorithms

The fastest algorithm in each case is the KNN algorithm using either the KD Tree or Ball Tree search algorithms (which search algorithm is quicker depends on the dataset, as shown in Table 2). Similarly, the slowest in each case is AdaBoost. The middle ground in each case is taken by the Random Forest models.

In contrast to speed, the accuracy ranking depends on the dataset. AdaBoost was the most accurate classifier for the Adult dataset, but at a significant time cost. AdaBoost had 0.7% better accuracy than the Random Forest classifier, but took almost six times the amount of time. This only amounted to about thirty seconds longer than the Random Forest classifier for testing predictions, but took several minutes longer during each round of 5-fold cross-validation. Thus, regarding tuning time, AdaBoost takes significantly longer than Random Forests.

One problem with Random Forest classifiers is their sensitivity to small changes in training data. For example, running Random Forest classifiers with different random seeds, but otherwise the same parameters, could alter their accuracy on the Ionosphere dataset from 78% to the 97% reported in Table 1. With this in mind, AdaBoost could be a better alternative to Random Forests when the dataset is small enough for the time cost to be prohibitive.

The KNN model was the fastest algorithm only because of the fast search algorithms implemented in the SciKit-Learn library. As demonstrated in Table 2, the KD Tree and Ball Tree search algorithms are much more computationally efficient than the Brute Force algorithm, with

Neighbors	Search Algorithm	Accuracy (%)	Training Time (s)	Prediction Time (s)
1	Brute	73.37%	0.024	23.5
3	Brute	75.60%	0.029	22.8
5	Brute	77.21%	0.039	20.5
7	Brute	77.98%	0.031	20.8
1	Ball Tree	73.39%	0.022	3.35
3	Ball Tree	75.60%	0.023	3.66
5	Ball Tree	77.20%	0.025	3.71
7	Ball Tree	77.98%	0.021	3.58
1	KD Tree	73.39%	0.021	2.79
3	KD Tree	75.60%	0.023	3.05
5	KD Tree	77.20%	0.023	2.90
7	KD Tree	77.99%	0.022	2.89

Table 2: KNN with varying search algorithm and number of neighbors

no cost in accuracy. Which was faster between KD Tree and Ball Tree depended on the dataset. The number of neighbors did not seem to affect the timing in any consistent manner. This may be due to the specific implementation of KNN, or to the small number of neighbors.

Conclusion:

It is very rare that a single algorithm does consistently better than others using several metrics, but in this experiment KNN seemed to be the most favorable. It was the quickest, whether training, tuning, or testing and was the most accurate classifier for two of the three datasets tested. The Random Forests classifiers are a bit slower, but consistently have high accuracy. The problem with Random Forests classifiers is their high variance in accuracy. Because of their sensitivity to small changes in training data, they can be tricky to use in practice. AdaBoost classifiers also have consistently high accuracy, but can take a prohibitively long time to tune on a laptop. If a quick model is needed, it seems that KNN is the go-to algorithm.

Bonus Points:

There are two places in this project where bonus points might be awarded. First, for the time taken to encode the Adult dataset using a one-hot encoding scheme manually. Second, for focusing on the time-efficiency of the algorithms, in addition to the accuracy of the algorithms.

References

Caruana, Rich and Alexandru Niculescu-Mizil. “An Empirical Comparison of Supervised Learning Algorithms.” 23rd International Conference on Machine Learning. 25 June 2006.