CS 280 Programming Assignment 2

Boosted Perceptrons and SVM

Isabelle Tingzon

2011-46060

November 11, 2015

Contents

1 Introduction

In the following programming assignment, I compare two state-the-art classification methods, namely Adaboost and Support Vector Machines. Adaboost combines weak learners to produce an ensemble of classifiers that acts as a strong learner. Perceptron Classifier is chosen as the weak learner and trained using the Pocket Algorithm. The resulting classifier is referred to as the *Boosted Perceptron*. Meanwhile, Support Vector Machines (SVM) are kernel-based learning algorithms that map non-linearly separable data into higher dimensional feature space where an optimal separating hyperplane can be optimally computed.

In the following sections, I present the results of training and testing the boosted perceptron on two different data sets (*banana data set* and *splice data set*) with up to K = 1000 learners. I also present the results of running the SVM Classifier on the said data sets. Finally, I compare the performance of SVM and Boosted Perceptrons in terms of accuracy and speed.

2 Implementation

Adaboost Algorithm was implemented in Python and tested against the Python implementation of SVM given in the SVM library `LIBSVM` [2]. Both algorithms were run on an Intel Core i5 processor running Ubuntu 14.04. To view the full source code of the boosted perceptron, please see the attached file, `boosted_perceptron.py`.

3 Performance Measures

3.1 Accuracy

The accuracy is given by the following.

$$\text{Accuracy} = \frac{\text{No. of correctly predicted data}}{\text{Total test data}} \times 100\% \tag{1}$$

Note that the accuracy of the boosted perceptron on a data set is the highest attained accuracy given K learners.

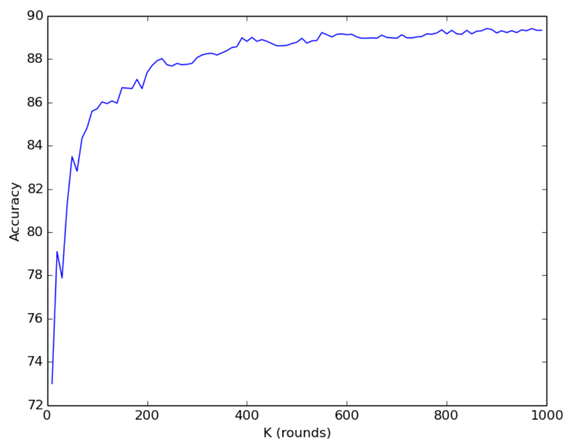
3.2 Speed

The speed is given by the time of execution of the python program which includes the time to train the algorithm on the training set as well as the time to classify the test set. To obtain the running time, I use the Unix utility `time` which returns the output measures `user` and `sys`. `user` gives the amount of time the CPU spent in user-mode code (outside of kernel) within the process, and `sys` gives the amounts of time the CPU spent in the kernel within the process. The total running time is taken as the sum `(user + sys)` which gives the actual CPU time the process used. [1]

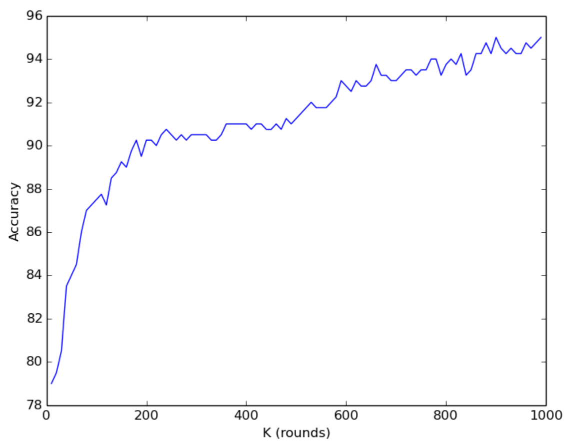
4 Results

4.1 Boosted Perceptron Plot Results

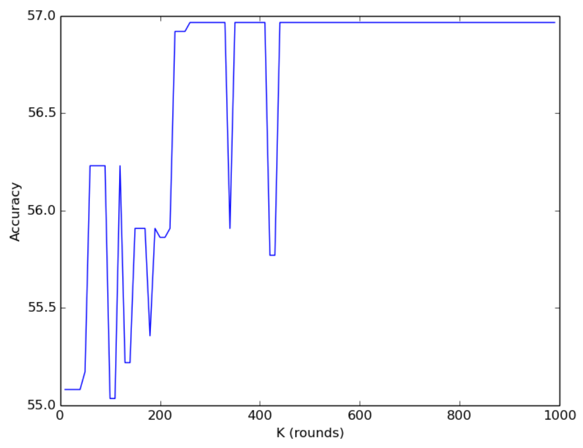
The following are the plots of the training and test accuracies of the banana and splice data sets against the number of learners K used for training where K = 10, 20, 30, $\cdots$, 1000. From the following results, accuracy appears to be increasing asymptotically with K.



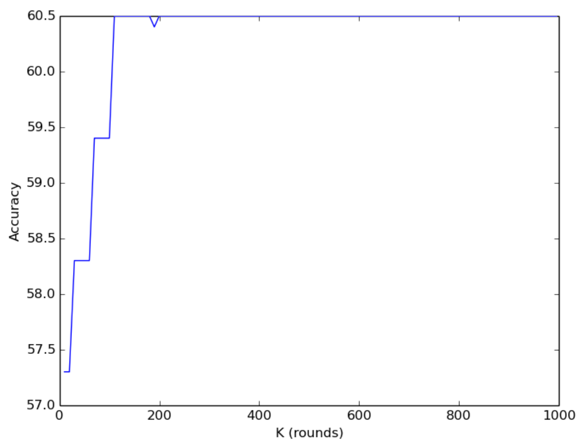
(a) banana test set



(b) banana training set



(a) splice test set



(b) splice training set

Figure 1: Plots of the training and test accuracies against the number of learners K

4.2 SVM Kernel and Parameter Settings

Obtaining the optimal performance using SVM involves choosing the best kernel and some fine tuning of the kernel parameters. The best kernel and kernel parameters for the banana and splice data set are as follows.

	Kernel	**Kernel Parameters**
banana data set	radial basis function $e^{-\gamma\|u-v\|^2}$	cost C = 72
splice data set	radial basis function $e^{-\gamma\|u-v\|^2}$	cost C = 10 degree d = 1

Table 1: Kernel Type and Kernel Parameters

4.3 Boosted Perceptron vs. SVM

	Boosted Perceptron		SVM	
	Highest Accuracy	Running Time	Accuracy	Running Time
banana training set	95%	7m46.946s	96.25%	0m0.063s
banana test set	89.41%	12m37.537s	66.12%	0m0.132s
splice training set	60.5%	8m43.301s	100%	0m0.588s
splice test set	56.97%	7m22.931s	92.41%	0m0.771s

Table 2: Comparison of SVM and Boosted Perceptron Performance

In terms of speed, it is easy to see that SVM trumps the boosted Perceptron by running in less than a second for all instances, whereas the boosted perceptron takes 10 minutes on average to finish its classification. We can attribute the expensive running time of the boosted perceptron to the fact that at each iteration from 1 to K, a weak learner is trained using the Pocket Algorithm (which performs 10000 iterations at maximum).

In terms of accuracy, the boosted perceptron and SVM both return relatively high levels of accuracy for the banana data set. For the banana training set, the highest accuracies attained by both algorithms are nearly equal (95% and 96.25%). For the banana test set, the boosted perceptron performs better than SVM by 35%. Meanwhile, for the splice data set (both training and test sets), SVM is the superior classifier with up to 100% accuracy for the training set and 92% for the test set. The boosted perceptron performs rather poorly for the splice test set with merely 57% which is only a little better than random guessing.

In conclusion, SVM outperforms the boosted perceptron in terms of speed. In terms of accuracy, SVM peforms better than the boosted perceptron for the splice data set but returns nearly the same level of accuracy for the banana data set.

References

[1] Python performance analysis. http://www.huyng.com/posts/python-performance-analysis/, 2015. Last accessed 11 November 2015.

[2] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.