Boosting Trees
Advanced Machine Learning

Francisco Calderon

2/3/2017

1. Adaboost on a toy set

1.1 Adaboost by hand

m	$w1$	$w2$	$w3$	$w4$	err	α	$Gm(x1)$	$Gm(x2)$	$Gm(x3)$	$Gm(x4)$
1	0.25	0.25	0.25	0.25	0.250	1.098612	-1	1	-1	-1
2	0.25	0.25	0.75	0.25	0.167	1.609438	-1	-1	1	-1
3	0.25	1.25	0.75	0.25	0.100	2.197225	1	1	1	-1
4	2.25	1.25	0.75	0.25	0.055	2.833213	-1	1	1	1

The trees trained where the following:

$G_1(x) = -1$ if $x_i^1 <= 0$ else 1
$G_2(x) = -1$ if $x_i^1 >= 0$ else 1
$G_3(x) = 1$ if $x_i^2 <= 0$ else -1
$G_4(x) = 1$ if $x_i^2 >= 0$ else -1

1.2 Error Calculation

1. $y = \{-1, 1, 1, -1\}$
2. $G_1(x) = \{-1, 1, -1, -1\}$
3. $G_2(x) = \{-1, -1, 1, -1\}$
4. $G_3(x) = \{1, 1, 1, -1\}$
5. $G_4(x) = \{-1, 1, 1, 1\}$

6. $G(x) = sign\big(\sum_{i=1}^{4}[1.0986 \cdot G_1(x_i) + 1.609 \cdot G_2(x_i) + 2.197 \cdot G_3(x_i) + 2.833 \cdot G_4(x_i)]\big)$
7. $G(x) = sign(\{-3.344039, 4.519612, 5.541264, -2.072061\})$
8. $G(x) = \{-1, 1, 1, -1\}$
9. $err = \frac{\sum \mathbb{I}(y \neq G(x))}{N} = 0/4 = 0$

1.3 Why Adaboost Is Better

The data set is not linearly separable. Each tree would need to split each variable more than once because you do not get pure labels from the just one split. AdaBoost performs better because it combines 4 decision stumps that at most have an error rate of .25, and each considers a different kind of split you could perform each feature. Then combining the strengths of each decision stump, the boosted trees can now perform better than one stump alone.

2.0 Implementing Adaboost

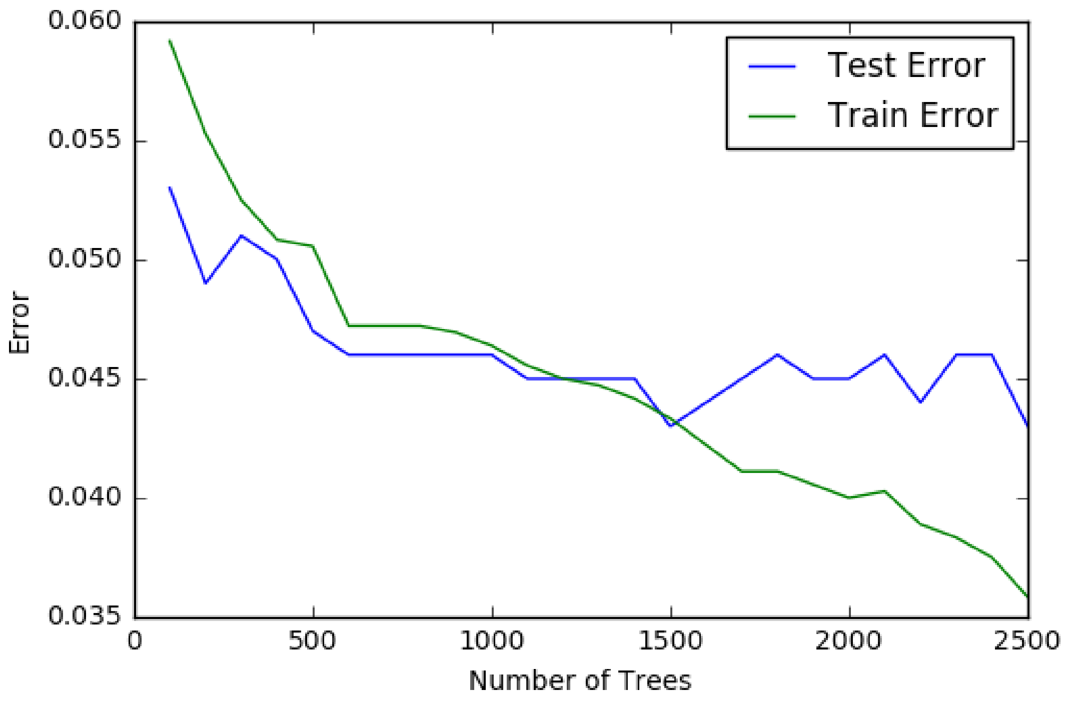

Figure 1: Train Error vs Test Error - AdaBoost

2.1 Tuning `numTrees`

In order to tune the `numTrees` parameter, the test data set is used as validation data to iterate over possible values of `numTrees` and find where we consistently good results for error. In this case, we can no longer use the test data set as an actual test set, but we can still optimize parameters such as `numTrees`. As shown in Figure 1, we find consistent results between 600 and 1000 trees. In order to keep the model simpler, a value between 600 and 700 trees would be the optimal place to set `numTrees`.

2.2 SciKit-Learn Algorithms

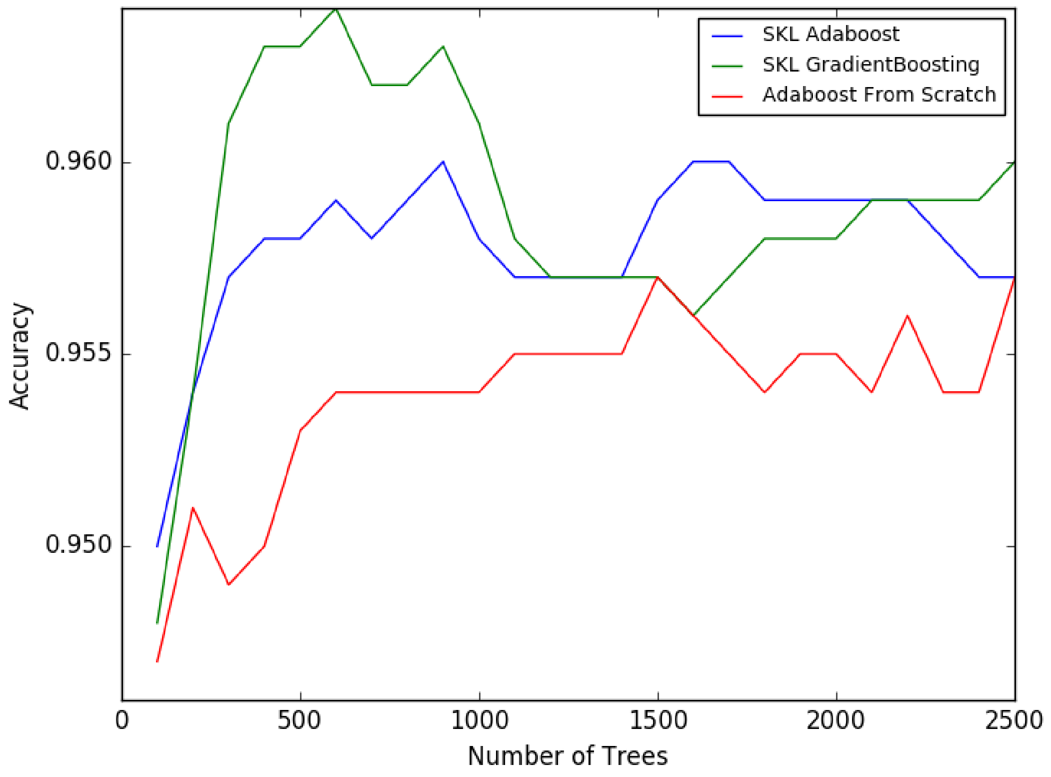

Figure 2: Battle of the Boosters

In Figure 2, the performance of each algorithm is measured as the number of trees or estimators increases. The ”homemade” `Adaboost` model performs admirably compared to SK-Learn’s `AdaBoost` and `GradientBoosting`. Both `AdaBoost` and `GradientBoosting` have parameters held constant with learning rate at 0.1 and max depth set to 1.

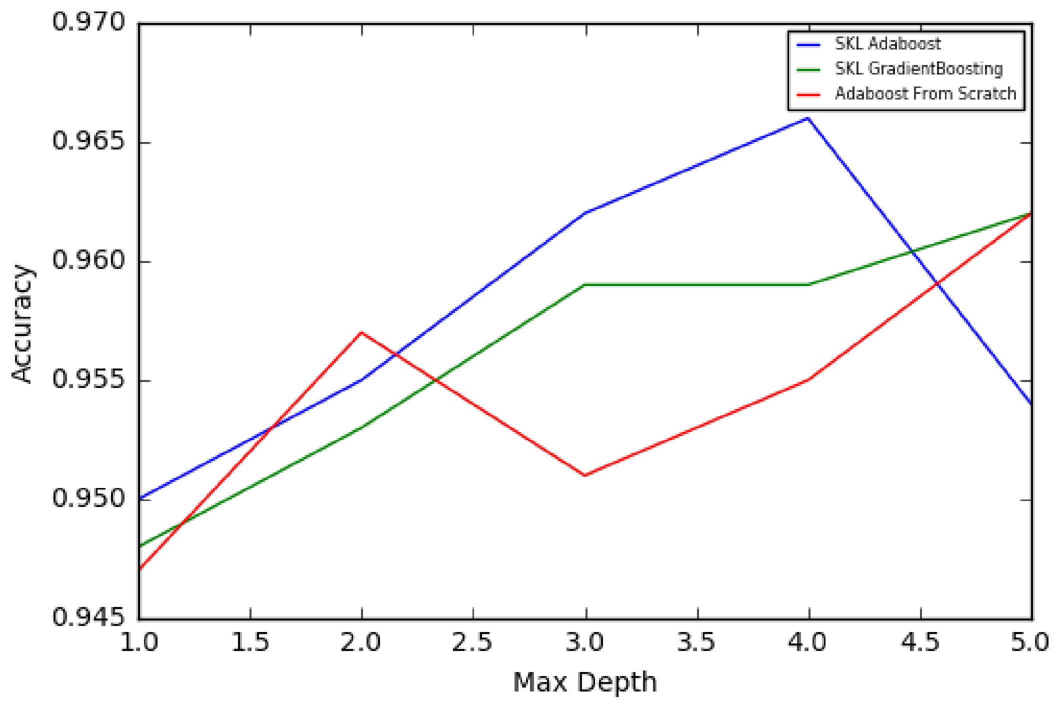

Figure 3: Battle of the Boosters

In Figure 3, the same algorithms are compared as before, but in this case the maximum depth increases from 1 to 5. `numTrees` is held constant for all models at 100 and learning rate is set to 0.1. The ”homemade” `Adaboost` is competitive with pre-packaged algorithms as the maximum depth increases.