Assignment 2 - Boosting

Shivakanth Thudi

2/4/2017

Part 1 - AdaBoost on toy dataset

1) AdaBoost iterations

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Figure 1: AdaBoost Algorithm

We implement the AdaBoost algorithm as described above in Excel and come up with the following trees. Note that each tree consists of just a stump, and the features we use at each step are shown in the Excel spreadsheet below.

SPLIT ON FEATURE_1 >= 0 (-1,1)									
Feature_1	Feature_2		Label		Weights		Predictions	Indicator		Weights*Indicator
0	-1		-1		0.25		-1	0		0
1	0		1		0.25		-1	1		0.25
-1	0		1		0.25		1	0		0
0	1		-1		0.25		-1	0		0
									Error	0.25
									Alpha	0.477121255
SPLIT ON FEATURE_2 >=0 (1,-1)									
Feature_1	Feature_2		Label		Weights		Predictions	Indicator		Weights*Indicator
0	-1		-1		0.25		-1	0		0
1	0		1		0.40285721		1	0		0
-1	0		1		0.25		1	0		0
0	1		-1		0.25		1	1		0.25
									Error	0.216852528
									Alpha	0.55767906
SPLIT ON FEATURE_1 <=0 (-1,1)									
Feature_1	Feature_2		Label		Weights		Predictions	Indicator		Weights*Indicator
0	-1		-1		0.25		-1	0		0
1	0		1		0.40285721		1	0		0
-1	0		1		0.25		-1	1		0.25
0	1		-1		0.4366535		-1	0		0
									Error	0.186635313
									Alpha	0.639291494
SPLIT ON FEATURE_2 <=0 (1,-1)									
Feature_1	Feature_2		Label		Weights		Predictions	Indicator		Weights*Indicator
0	-1		-1		0.25		1	1		0.25
1	0		1		0.40285721		1	0		0
-1	0		1		0.47378442		1	0		0
0	1		-1		0.4366535		-1	0		0
									Error	0.15991862
									Alpha	0.720422325

Figure 2: Modeling Iterations

FINAL MODEL						
Feature_1	Feature_2		Label		Weighted Predictions from AdaBoost	
0	-1		-1		-1	
1	0		1		1	
-1	0		1		1	
0	1		-1		-1	
				Training Error	0	

Figure 3: Final Model

m	w_1	w_2	w_3	w_4	err	α	$G_m(x_1)$	$G_m(x_2)$	$G_m(x_3)$	$G_m(x_4)$
1	0.25	0.25	0.25	0.25	0.25	0.47712125	-1	-1	1	-1
2	0.25	0.40285721	0.25	0.25	0.21685253	0.55767906	-1	1	1	1
3	0.25	0.40285721	0.25	0.4366535	0.18663531	0.63929149	-1	1	-1	-1
4	0.25	0.40285721	0.47378442	0.4366535	0.15991862	0.72042233	1	1	1	-1

Figure 4: Summary

2) Training Error

The training error of AdaBoost for this toy dataset is 0.0. The AdaBoost model makes predictions using the weighted sum of the trees, and the computation is shown below:

FINAL MODEL											
Feature_1	Feature_2	Label	Tree1 predictions	α	Tree2 predictions	α	Tree3 predictions	α	Tree4 predictions	α	Weighted Predictions
0	-1	-1	-1	0.477	-1	0.558	-1	0.639	1	0.7204	-1
1	0	1	-1	0.477	1	0.558	1	0.639	1	0.7204	1
-1	0	1	1	0.477	1	0.558	-1	0.639	1	0.7204	1
0	1	-1	-1	0.477	1	0.558	-1	0.639	-1	0.7204	-1
											Error 0

Figure 5: Computations

3) Linearly Separable

Yes, the above dataset is linearly separable. Here, each decision tree incorrectly classifies one of the observations, but the weighted sum of the trees correctly classifies all observations.

A decision stump is a weak learner, and the AdaBoost algorithm increases the performance of weak classifiers such as decision stumps by reweighting observations in each iteration to focus on those observations that were misclassified in the previous modeling step. Each successive classifier concentrates on those training observations that were missed by previous ones in the sequence. The final AdaBoost model is a weighted sum of all the models that were used, and performs better than how each of the models would have performed individually.

Part 2 - Implementing AdaBoost

1) Training and Test Accuracy using AdaBoost with 100 trees

Training Accuracy with 100 trees	Testing Accuracy with 100 trees
0.9408	0.9470

2) Best value of numTrees - 1900 trees

We find the best value of numTrees by performing cross-validation on the training set for different values of numTrees and selecting the value that gives the highest cross-validation accuracy.

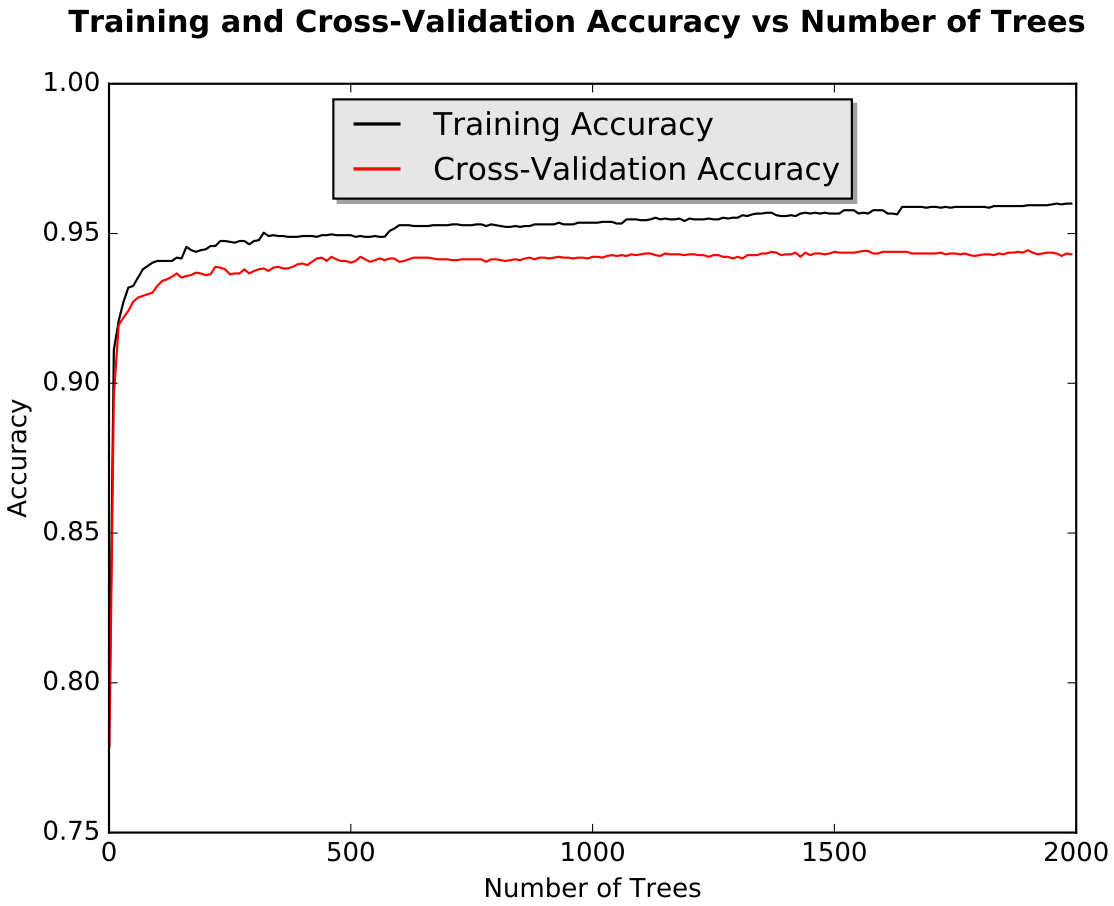

Figure 6: Plot of Accuracies using different values of numTrees

For the number of estimators (trees), I looked at values ranging from 1 to 2000 trees. The best cross-validation accuracy was 0.9444 with 1900 trees. However, from the plot, we see that the gains in accuracy are very marginal when we increase the number of trees beyond 1000.

We will pick 1900 as the number of trees since that gave the highest cross-validation accuracy.**We find that the accuracy on the test set using 1900 trees is 0.9550.**

Training Accuracy using 1900 trees	Testing Accuracy using 1900 trees
0.9594	0.9550

3) - Comparing AdaBoost with Gradient Boosting

For Gradient Boosting, the hyper-parameters we choose to tune are the number of estimators (trees), the maximum depth of each tree, and the learning rate. We consider a wide range for these parameters, and use the GridSearchCV library from sklearn.

Number of Estimators

We check values ranging from 50 to 450 with a step size of 50 for the number of trees. We see that 400 trees is the optimal value with cross-validation.

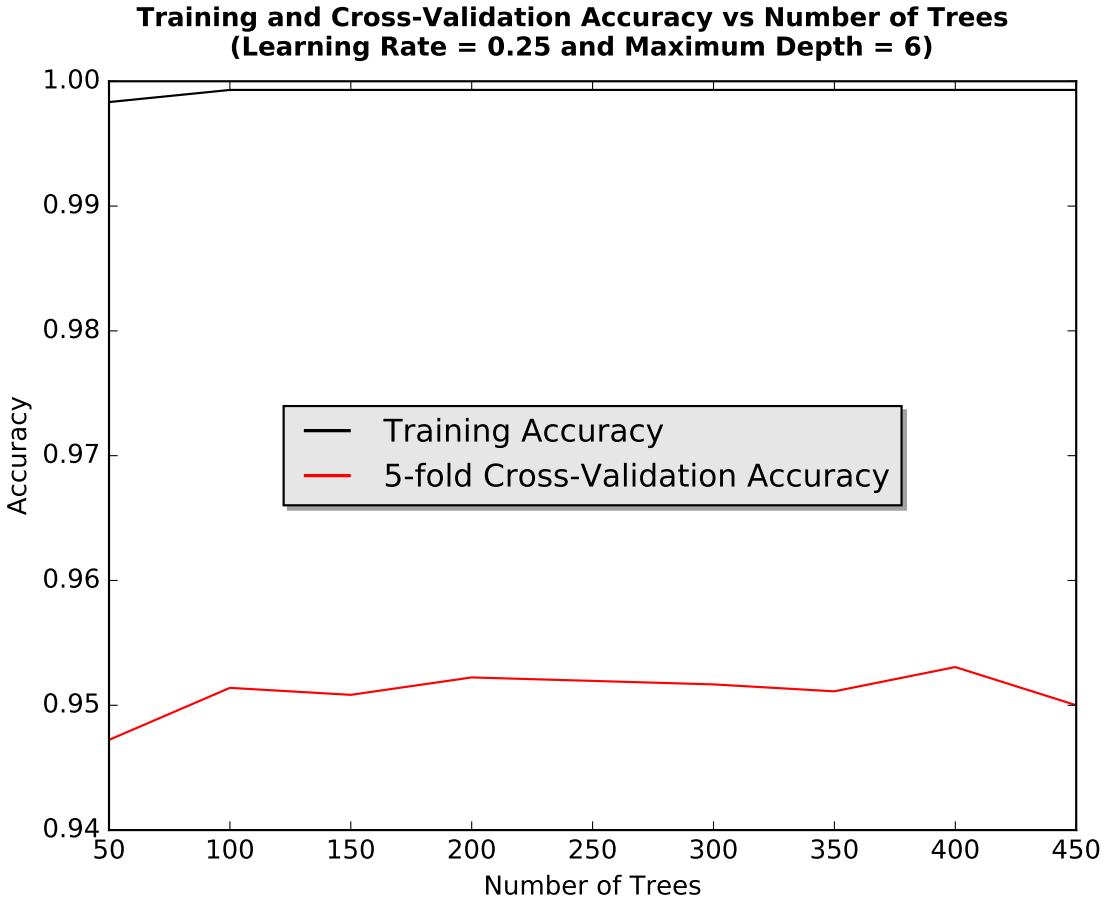

Figure 7: Plot of Accuracies vs Number of Estimators

Learning Rate

We consider learning rates of 0.05, 0.1, 0.15, 0.2, 0.25, and 0.30. We see that the optimal learning rate was 0.1.

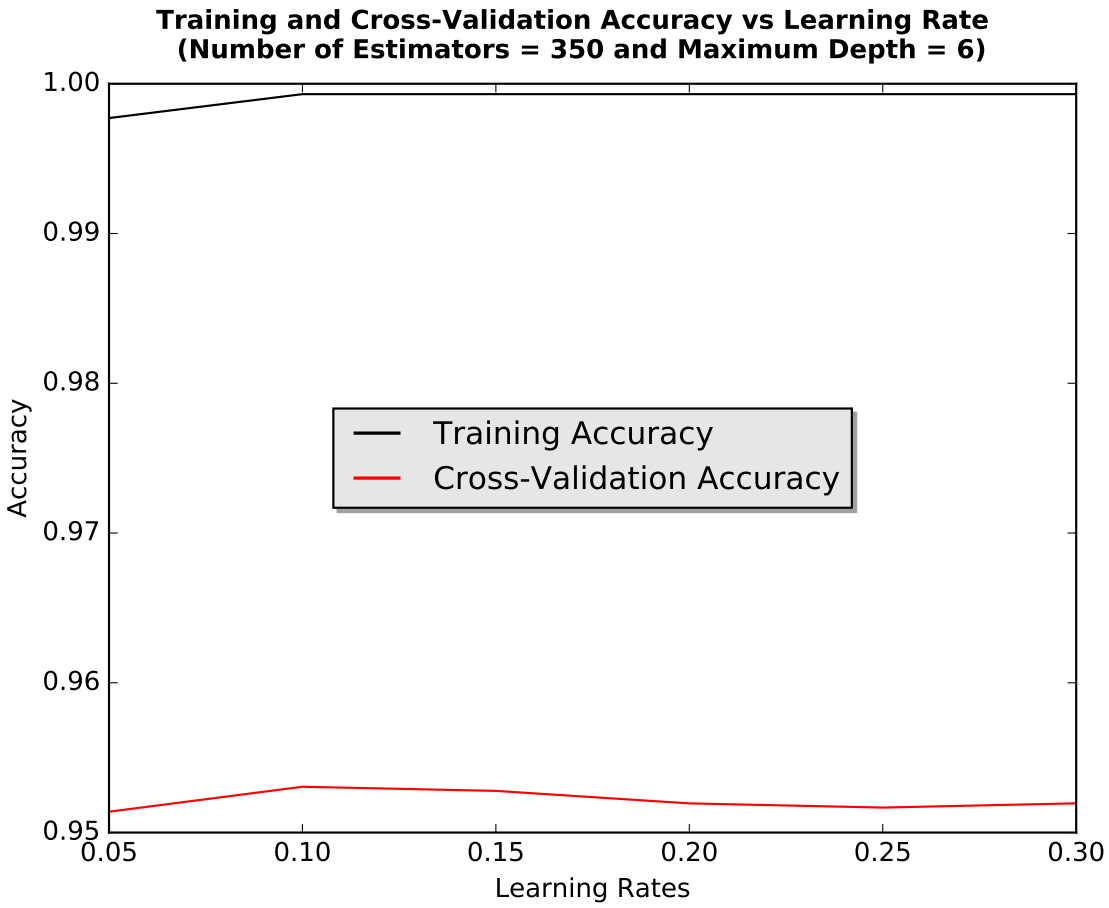

Figure 8: Plot of Accuracies vs Learning Rates

Maximum Depth of Each Estimator

We check values ranging from 1 to 10 for the depth of the tree. The optimal depth was found to be 7 trees.

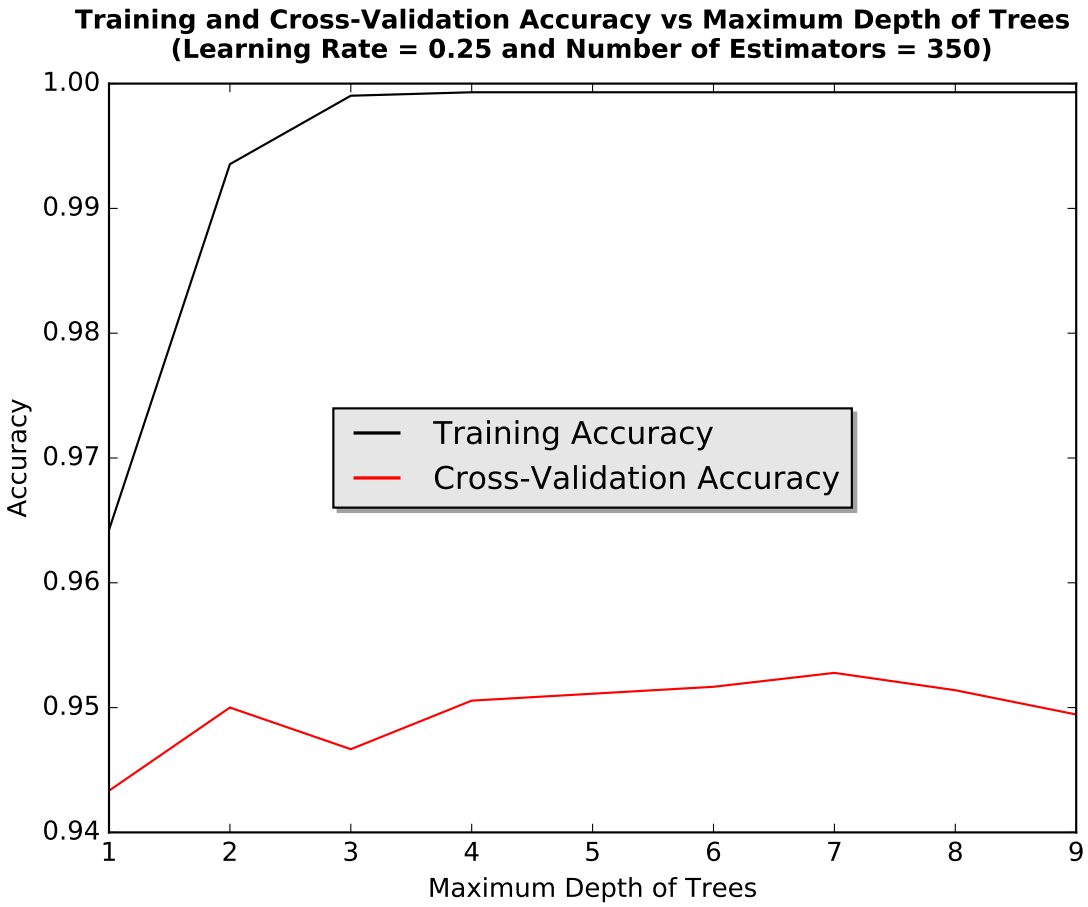

Figure 9: Plot of Accuracies vs Maximum Depth of Estimators

Optimal Combination of Hyper-Parameters

When we perform a grid search over all the possible combinations we took, we find that the optimal combination of hyper-parameters was the following:

- Number of Estimators = 400
- Maximum Depth = 6
- Learning Rate = 0.1

Comparison with AdaBoost

With the optimal combination of parameters, we found that Gradient Boosting performs better on the test set than AdaBoost. The table below summarizes these results:

Algorithm	5-fold Cross-Validation Accuracy	Training Accuracy	Testing Accuracy	Optimal Hyper-Parameters
AdaBoost	0.9444	0.9594	0.9550	1900 trees
Gradient Boosting	0.9547	0.9991	0.967	400 estimators, maximimum depth of 6 and learning rate of 0.1

The test accuracy using AdaBoost was 0.9550 , while it was 0.967 with Gradient Boosting.