Andrew Smith — adsmith@andrew.cmu.edu
10-601 — Machine Learning

(I did not collaborate with anybody else on this assignment.)

1 RandomForest

1.1 Background

The Random Forest classifier is an ensemble learning method that uses several Random Tree classifiers. Each tree is trained by projecting the data into a randomly chosen subspace before randomly fitting a decision tree to the lower-dimensional data. In classifying, all of the trees vote on the final classification.

Random Forest was introduced by Leo Breiman in 1999.

1.2 Parameters

1.2.1 Number of Trees

Increasing this parameter should decrease error but increase learning time and classifier size. It also may lead to overfitting.

Cross-validation suggests that around 25 trees is the best setting for this parameter. Running cross-validation dynamically, however, is not feasible due to the increase in runtime as this parameter increases.

1.2.2 Number of Features

This is the dimension of the random subspace into which the data is projected. Increasing this makes the individual trees' decisions stronger, but increases correlation between them.

The optimal results for this seem to vary very much with which dataset is trained on. Therefore this should be dynamically selected.

1.2.3 Depth of Trees

Increasing the depth of the individual trees may make them stronger, but limiting the depth may reduce overfitting.

Running this on the data makes it very clear that the trees are much stronger when their depth is not capped. Running with cross-validation almost always chooses this as well.

1.3 Final Settings

For my final settings, I chose fixed parameters of 25 trees and unlimited depth, but to use internal cross-validation to select the number of features. Due to time constraints, I am only performing 5-fold cross-validation.

2 VotedPerceptron

2.1 Background

The Voted Perceptron classifier is a modification of the perceptron classifier. Like perceptron, it learns a linear boundary by repeatedly adding any incorrectly classified data points in the appropriate direction. However, unlike a regular perceptron it stores each boundary it considers, along with how many data points were correctly classified by that boundary. The final boundary is then a weighted sum of the considered boundaries.

The perceptron classifier was introduced by Frank Rosenblatt in 1958. The voted perceptron modification was introduced by Yoav Freund and Robert Schapire in 1999.

2.2 Parameters

2.2.1 Iterations

This is the number of times the perceptron passes through the data. Increasing the number of iterations linearly increases classification time as well as the size of the classifier, but generally decreases training error.

This parameter has only a small performance increase, and seems to plateau at around 20-30. Due to time constraints, I fixed it at 25.

2.2.2 Alterations

This is a cap on the number of times the classifier will modify its decision boundary. Increasing it should behave similarly to increasing the iterations, until it is large enough that it has no effect anymore.

The voted perceptron classifier is very robust to changes in this parameter. For higher numbers of iterations, though, decreasing the alterations drastically decreases the learning time. After experimentation I fixed this at 500.

2.2.3 Exponent

This is the exponent of the polynomial kernel used in the kernel trick to linearize the data. Increasing this should decrease training error, but increase learning time, classifier size, and overfitting.

Increasing this leads to overfitting very quickly. Cross-validation gives 2 as the best setting for this.

2.3 Final Settings

My final settings for this classifier uses fixed settings for all three settings, as the main concern for two of them is time, and a higher exponent is likely to always be too overfit. The parameters are fied at 25, 500, and 2. On the training data, this takes about 8 seconds to train all 12 datasets.

3 LogitBoost

3.1 Background

LogitBoost is based on the earlier AdaBoost algorithm, which is an ensemble method producing a linear combination of other classifiers. LogitBoost uses the AdaBoost algorithm to minimize the cost function from logistic regression.

LogitBoost was introduced by Jerome Friedman, Trevor Hastie, and Robert Tibshirani in 2000. AdaBoost was introduced by Freud and Schapire in 1995.

3.2 Parameters

3.2.1 Shrinkage

This is a regularization parameter in the AdaBoost algorithm. It can decrease overfit.

In testing, any value less than 1 for this parameter makes performance much worse.

3.2.2 Iterations

This controls how long AdaBoost is allowed to converge. Increasing it should improve training error. Decreasing it should improve learning time and possibly reduce overfitting.

Cross-validation shows that this is almost always chosen to be 10.

3.3 Final Settings

LogitBoost does not seem to have much room for tuning its parameters. No changes away from the default parameters proved beneficial.

4 Summary

With the default settings LogitBoost was the most robust classifier, with an average $error_C/error_{NB}$ of 0.73 and max of 1.04. However, LogitBoost proved resistant to parameter tuning, while RandomForest benefited from improved settings to two parameters and internal cross-validation on another. This improved the average $error_C/error_{NB}$ of RandomForest from 0.75 to 0.68, and the maximum from 1.21 to 1.05.