CS 543 Programming assignment

SEIFEDDINE MEJRI (33%)

WAJID RAMEEZ (33%)

ADHIKARI AASHISH (33%)

NB: In this report we often use test instead of validation. We mean by test the same as Validation.

Part1:

(a)

Solution

A decision tree using a maximum depth of 20 was created as illustrated in the code.

NB: Part1.py contains the final version of the implementation.

(b)

Solution

Following is the plot obtained by plotting the training and validation accuracies against the depths of the trees.

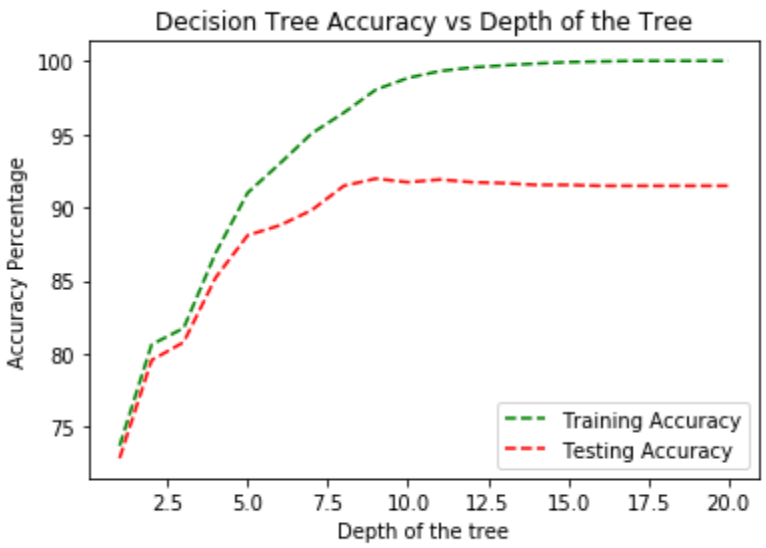

Figure 1 Train and validiation accuracy versus depth

(c)

Solution

As observed in the plot, the training accuracy increases as the depth of the tree we create increases.

At the depth of 17, it reaches the accuracy of 100%. This is because with each increase in depth, we divide the training examples more and more on the basis of the benefit of split. Since the tree was itself generated from the given examples, it is obvious that as we increase the depth of the tree, it is going to perform well on the training data. In general, this eventually leads to overfitting the model to the given training data. Overfitting a model leads to a bad accuracy on the testing data. As I can be seen on the curve above, the validation accuracy decreases after reaching a maximum value because of overfitting on the training data.

Our tree reached the accuracy of 100% on the training data at the depth of 17.

(d)

Solution

The best validation accuracy was found at the depth of 9 and it was 91.95%. After that, the accuracy started decreasing because of overfitting.

Part2:

(a) Random Forest algorithm was implemented as stated in the instructions. Please see the code under part2.py.

(b) The plot for train and validation accuracy is as follows:

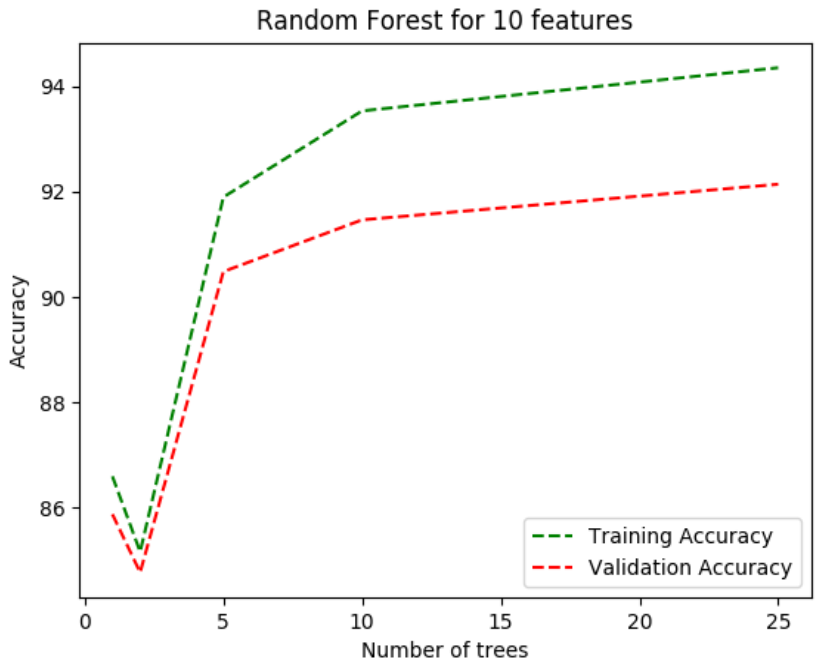

Figure 2Train and validation accuracy of the forest versus the number of trees

(c) Adding more trees generally have a positive effect on the accuracy as more trees are now voting for the correct prediction. Also, error of the average of the prediction of the tree ensemble will always be smaller or equal than the average error of individual trees.

(d) The plots for greater number of ‘m’ (20,50) are as the following:

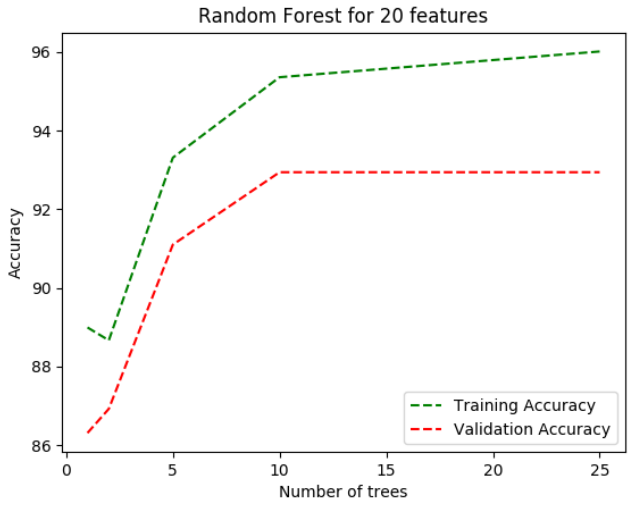

Figure 4 Acc with different number of trees for m =20

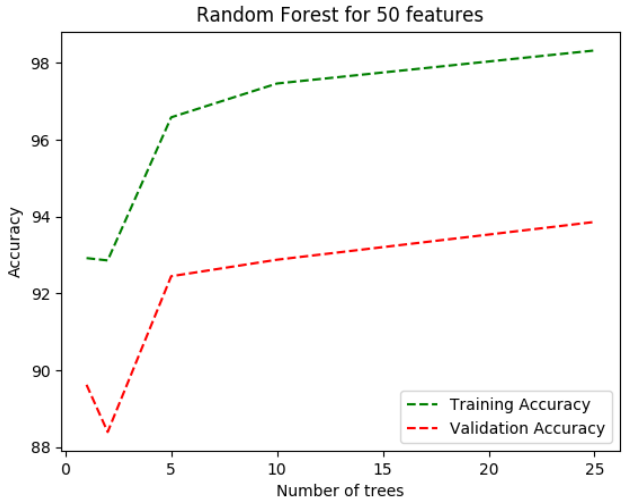

Figure 3 Acc with different number of trees for m =50

Adding more features to Random Forest result in a higher training accuracy. But the validation accuracy remains nearly the same. This may be due to more feature resulting in a more complex model and going towards over-fitting on the training set.

Part3:

(a)

Solution

The weak learner decision tree with depth=9 was implemented with the given requirements.

The code of the week learner is under DT_weak_learner.py

(b)

Solution

The Adaboost was implemented with the number of learners as [1, 5, 10, 20].

(c)

Solution

Training and Validation Accuracy for the Learners:

Number of Learners, L	Training Accuracy	Validation Accuracy
1	0.99	0.90
5	1	0.93
10	1	0.94
20	1	0.96

Table-1training and validation accuracy using all example

Number of Learners, L	Training Accuracy	Validation Accuracy
1	1	0.69
5	1	0.78
10	1	0.81
20	1	0.83

Table-2 Training and validation accuracies with only 100 examples used.

(d)

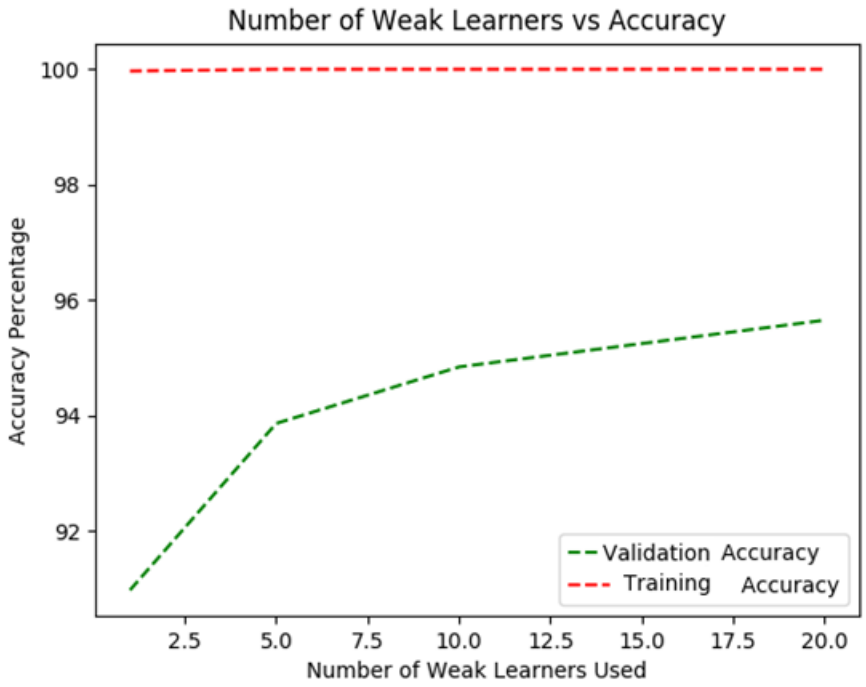

Figure 5 Training and validation accuracies (Ada) using all training examples for building each tree

The training accuracy is 99% for L =1 and depth = 9. (Note: Had we used a deeper tree, it would have eventually completely overfitted to the training data as we saw in part 1.) As we increase the number of learners that contribute to the final vote, the training accuracy reaches 100% for all values L =[5, 10, and 20]. Indeed, since more learners get to vote, a training example misclassified by one learner is overshadowed by the combined votes of other learners.

In case of the validation data, we see that the validation accuracy is a little less than 91% for L = 1. This accuracy increases with the increase of learners because more as the number of learners that vote increases, the number of correct votes on a training example also tends to increase i.e., different classifiers are less likely to make the same mistake and thus this principle leads to correct predictions on many validation examples.