CS534 — Implementation Assignment 3

Chih-Hsiang Wang
Jui-Hung Lu
Webster Cheng

Contribution	Chih-Hsiang Wang	Jui-Hung Lu	Webster Cheng
Percentage (%)	33.3	33.3	33.4

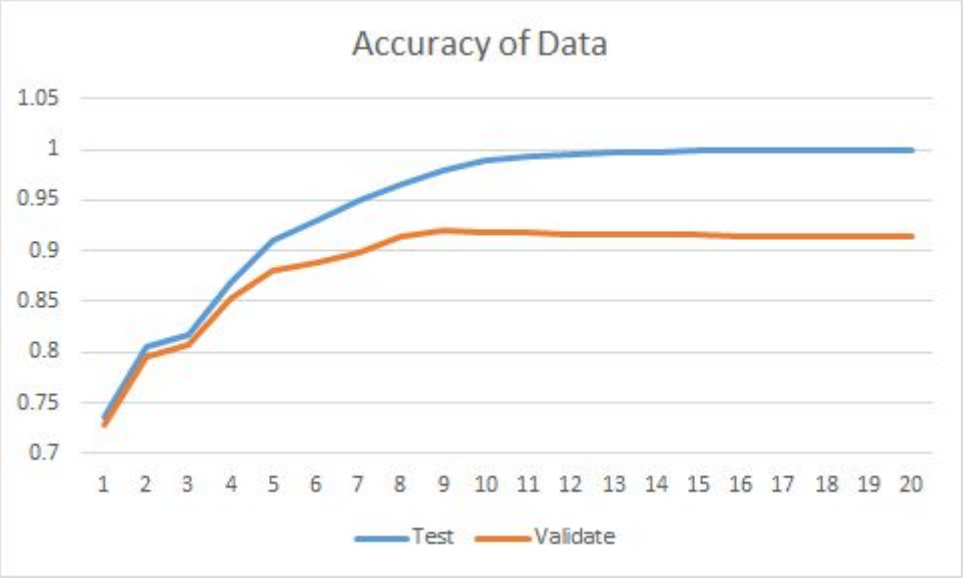
General Introduction: In this report, we implemented decision tree, random forest, and AdaBoost. The later two were implemented based on the decision tree in part 1.

Part 1 (20 pts) : Decision Tree (DT). Please implement below steps:

(a) Create a decision tree with maximum depth of 20 (root is at depth=0) on the train data. (Note that a normal implementation of the tree should take about 220 seconds on Babylon for this maximum of depth.)
(implemented in “decisionTree.py”)

(b) Using the created decision tree, compute and plot the train and validation accuracy versus depth.

Depth	Train Accuracy	Validation Accuracy
1	0.737111293	0.728667894
2	0.806260229	0.79558011
3	0.817512275	0.807857581
4	0.86886252	0.852056476
5	0.909983633	0.880908533
6	0.929828151	0.887661142
7	0.950490998	0.898096992
8	0.964607201	0.914671578
9	0.980360065	0.919582566
10	0.988338789	0.918354819
11	0.99304419	0.918354819
12	0.995499182	0.917127072
13	0.99693126	0.916513198
14	0.998158756	0.915899325
15	0.999181669	0.915285451
16	0.999590835	0.914671578
17	1	0.914671578
18	1	0.914671578
19	1	0.914671578
20	1	0.914671578



(c) Explain the behavior of train/validation performance against the depth. At which depth the train accuracy reaches to 100% accuracy? If your tree could not get to 100% before the depth of 20, keep on extending the tree in depth until it reaches 100% for the train accuracy.

As the depth increases, the accuracy of training increases that it will eventually reach to 100% no matter the data is. Afterward, the accuracy of both training and validation will remain the same.

In our model, the accuracy reaches to 100% when the depth is 17.

(d) Report the depth that gives the best validation accuracy?

When depth is 9, the validation accuracy is the best.

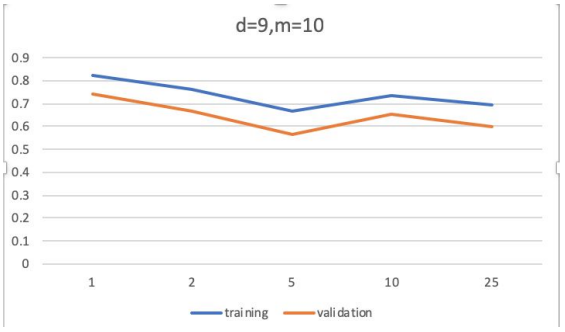
Depth	Train Accuracy	Validation Accuracy
9	0.980360065	0.919582566

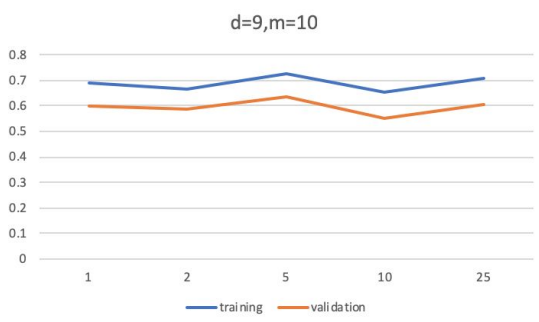
It refers that we need to stop the growing of the tree by ourselves, or the tree will be overfitting.

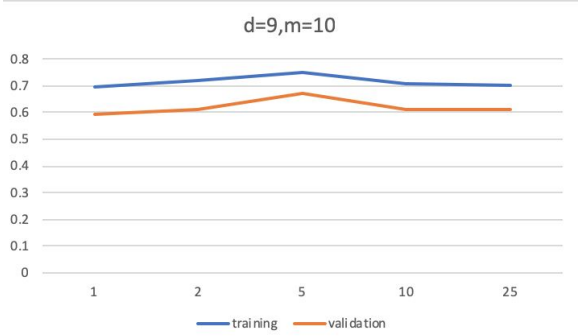
Part 2 (30 pts) : Random Forest (Bagging). In this part we are interested in random forest which is a variation of bagging without some of its limitations. Please implement below steps:

(a) Implement a random forest with below parameters:
n : The number of trees in the forest.
m : The number of features for a tree.
d : Maximum depth of the trees in the forest.
Here is how the forest is created: The random forest is a collection of n trees. All the trees in the forest has maximum depth of d. Each tree is built on a data set of size 4888 sampled (with replacement) from the train set. In the process of building a tree of the forest, each time we try to find the best feature fi to split, we need to first sub-sample (without replacement) m number of features from 100 feature set and then pick fi with highest benefit from m sampled features.
(implemented in “randomForest.py”)

(b) For d = 9, m = 10 and n ∈ [1, 2, 5, 10, 25], plot the train and validation accuracy of the forest versus the number of trees in the forest n.



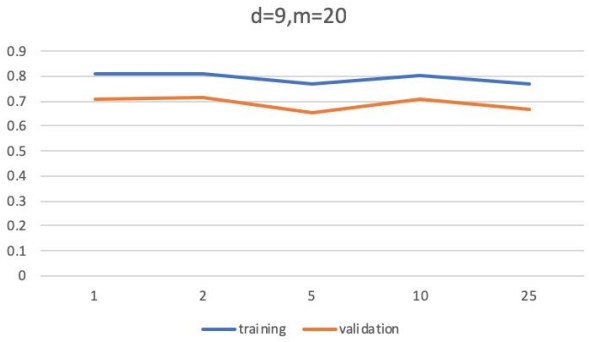


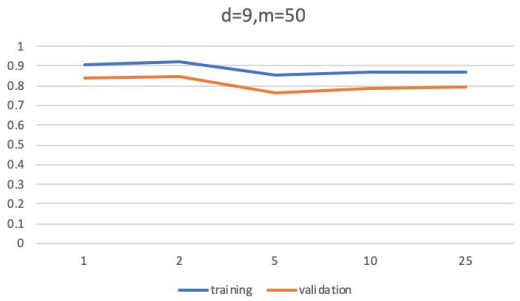


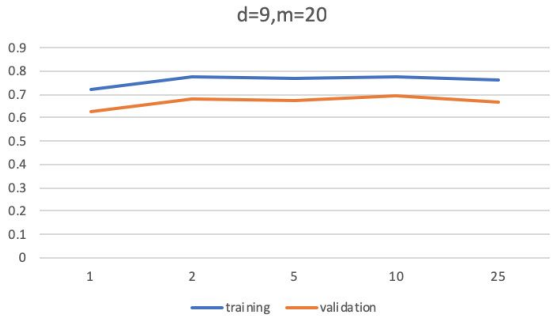
(c) What effect adding more tree into a forest has on the train/validation performance? Why?

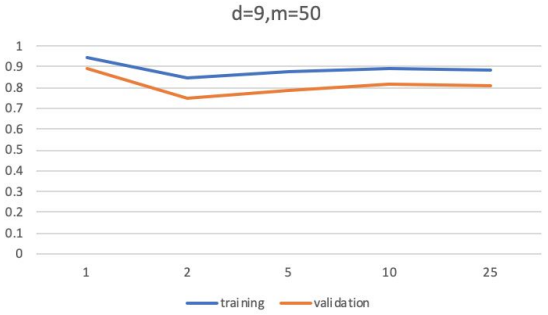
When the number of tree in a forest increases, the accuracy will be more stable. Because we can reduce the variance by voting from each tree. However, it is not equal to having higher accuracy when number of tree increases.

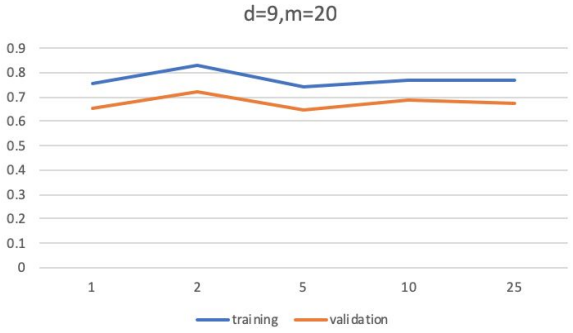
(d) Repeat above experiments for d = 9 and m ∈ [20,50]. How greater m changes the train/validation accuracy? Why?

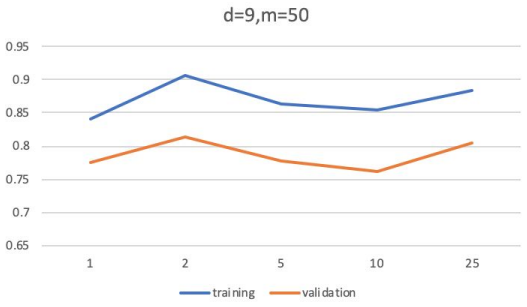












When the number of sub-sample of features increases, the accuracy will increase. Because each tree has more features to get better Gini index for dividing branches, the accuracy for both training and validation will increase.

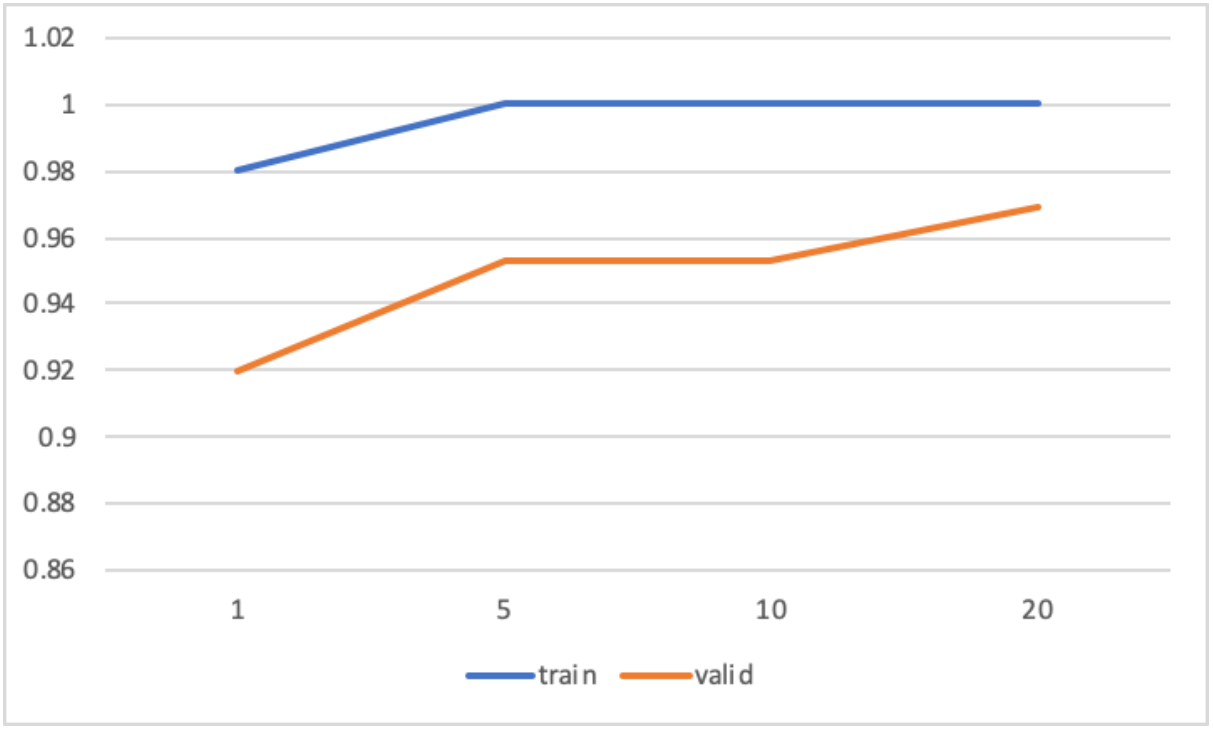
Besides, according to the expected loss formula:
expected loss = (bias)^2 +variance+noise
When we randomly choose the datas into our trees, we reduce the variance.

Part 3 (30 pts) : AdaBoost (Boosting). For this part we are interested in applying AdaBoost to create yet another ensemble model with decision tree. Considering the AdaBoost algorithm described in the slide, please do below steps:

(a) Let the weak learner be a decision tree with depth of 9. The decision tree should get a weight parameter D which is a vector of size 4888 (train size). Implement the decision tree with parameter D such that it considers D in its functionality.
(Hint: It changes the gini-index and also the predictions at leaves).
(implemented in “decisionTree.py”)

(b) Using the decision tree with parameter D implemented above, develop the AdaBoost algorithm as described in the slide with parameter L.
(implemented in “adaboost.py”)

(c) Report the train and validation accuracy for L ∈ [1, 5, 10, 20].



(d) Explain the behavior of AdaBoost against the parameter L.
When the parameter L increases, the accuracy will increase. To be noticed, even when training accuracy has already achieved one, the validation accuracy can still increase for more iteration of AdaBoost.

During each iteration of AdaBoost, we increase the weight of the data if the data is misclassified, otherwise, we decrease the weight. As a result, we enable weak classifier to concentrate on previously misclassified data and improve the overall accuracy.