(CS-534) Machine Learning
Implementation Assignment #3

Decision Tree Ensemble for Optical Character Recognition

Submission By:

Akash Agarwal
OSU ID NUMBER: 933-471-097
Vishnupriya Nochikaduthekkedath Reghunathan
OSU ID NUMBER: 933-620-571
Aashwin Vats
OSU ID NUMBER: 933-615-112

Contribution by each member in percentage : Equal contribution by each member

Akash Agarwal - 33%
Vishnupriya Nochikaduthekkedath Reghunathan - 33%
Aashwin Vats - 33%

Introduction

In this assignment, we are developing variations of decision trees to classify handwritten digits of numbers 3 and 5. Here we are using test and validation data to study the behavior of variations in decision trees.

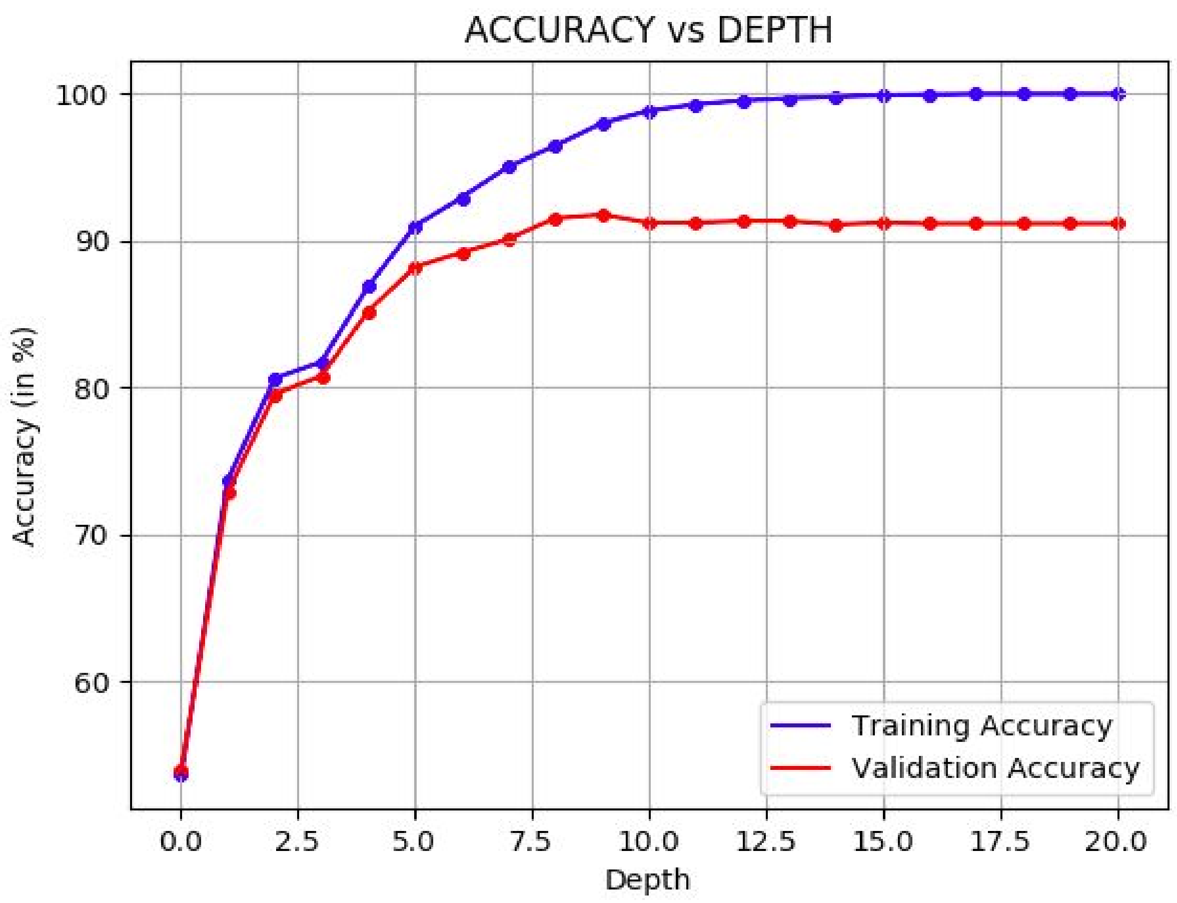
Preprocessing
The output Y from the train and validation datasets are changed to 1 and -1 (+1 to 3 and -1 to 5) and removed from the dataset for further processing with the train and validation data.

Part 1 : Decision Tree (DT)

In this part we implemented a decision tree that uses gini index to measure the uncertainty. We created the Decision tree with maximum depth 20 and root node at depth 0 on train data. Then we used the decision tree to classify train and validation data. Train and validation accuracy vs depth is recorded.

A. **Create a decision tree with maximum depth of 20 (root is at depth=0) on the train data.**
   Solution: The code for building a decision tree of maximum depth D was implemented. The program takes approximately 200 secs to build a decision tree with maximum depth 20.

B. **Using the created decision tree, compute and plot the train and validation accuracy versus depth.**
   Solution:



C. **Explain the behavior of train/validation performance against the depth. At which depth the train accuracy reaches to 100% accuracy?**
Solution: The model reaches 100% training accuracy at **depth 17** (depth starts at 0), and it remains constant for higher depth. But **on validation data the model does not reach 100% accuracy**. The maximum accuracy it reaches is **91.77 % at depth 9**. After that depth the accuracy slightly drops and it reaches 91.528% accuracy at depth 20.

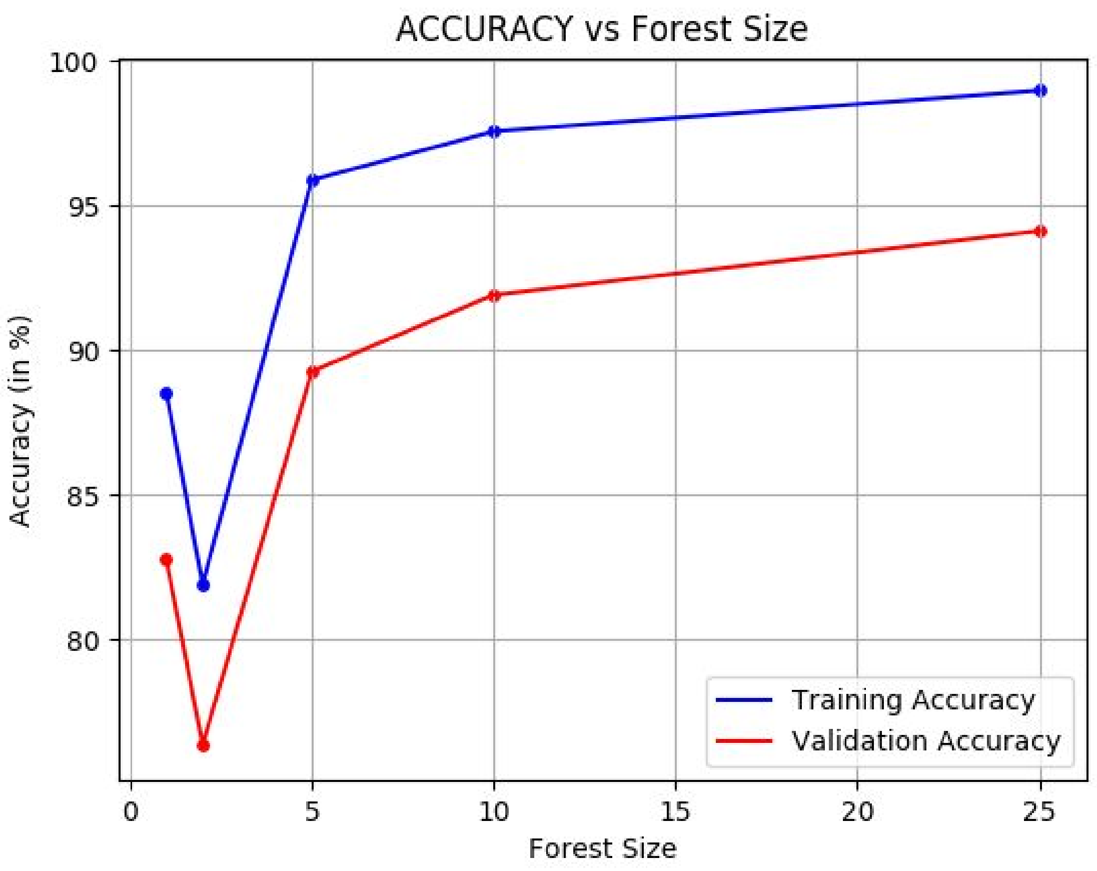
D. **Report the depth that gives the best validation accuracy?**
Solution: The best validation accuracy is **91.77 % at depth 9**.

Part 2: Random Forest (Bagging)

In this Part, we are creating a forest of multiple decision trees, that choose 'm' features out of total 100 features, to calculate the maximum information gain at each node of the tree. Each tree in the forest randomly selects m features. Then we collect the data obtained and analyze it to answer the questions that are being asked. The same decision tree creation function used in Part-1, is called iteratively to create the whole forest.

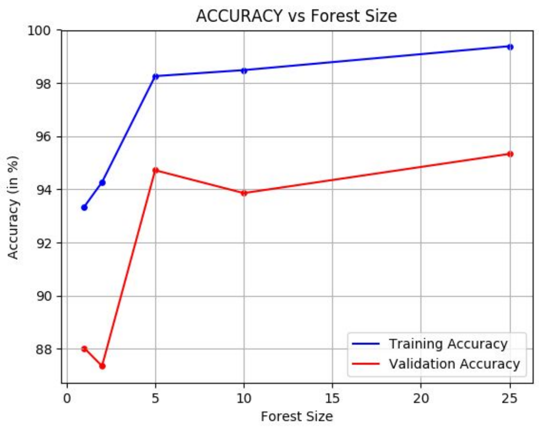
A. **Implement a random forest with below parameters:**
n : The number of trees in the forest.
m : The number of features for a tree.
d : Maximum depth of the trees in the forest.
Here is how the forest is created: The random forest is a collection of n trees. All the trees in the forest has maximum 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 f_i to split, we need to first sub-sample (without replacement) m number of features from 100 feature set and then pick f_i with highest benefit from m sampled features.
Solution: Random Forest implemented.

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.**
Solution:

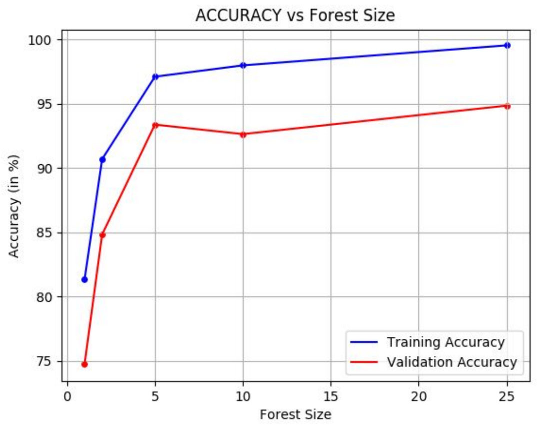


C. **What effect adding more tree into a forest has on the train/validation performance? Why?**
Solution: The accuracy will be increased when we add more trees into the forest for both training and validation accuracy. The output is taken by majority vote in random forest so any biases will be cancelled out. It was observed that after we increase the size of forest from 10 to 25, the validation accuracy increases very slightly, whereas, training accuracy increases more rapidly.

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



m = 50



m = 20

It can be observed from the graphs, the greater the value of m, i.e. the number of features selected randomly to calculate information gain, the more will be the training and validation accuracy for forests with less number of trees. It should also be noticed as we increase the number of trees in the forest, the training and validation accuracies become approximately same for different values of m.

This is because, more m means a better learner and vice versa. Thus, a forest with less number of trees with greater m will have better accuracy as compared to forest with same number of trees but lower value of m. As the forest size increases, due to bagging the weak learners combine their work to form a strong learner, hence for forests with large number of trees, the value of m do not have any significant effect on the training and validation accuracy.

Part 3: AdaBoost (Boosting)
In this part, we are using a Boosting method, known as Adaptive Boosting (AdaBoost) to create the decision tree classifier. Adaboost makes use of L Weak Learners to create a strong classifier by making use of Distribution matrix D, that contains weights corresponding to each training example. We train the decision tree calculating errors at each iteration and changing the corresponding weights of mis-classified examples to improve the accuracy for the next learner, thus, increasing the accuracy of the aggregated classifier.
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).**
Solution: The Adaboost algorithm was implemented. It takes in account the distribution matrix D and calculates gini-index and predictions while considering D's functionality.

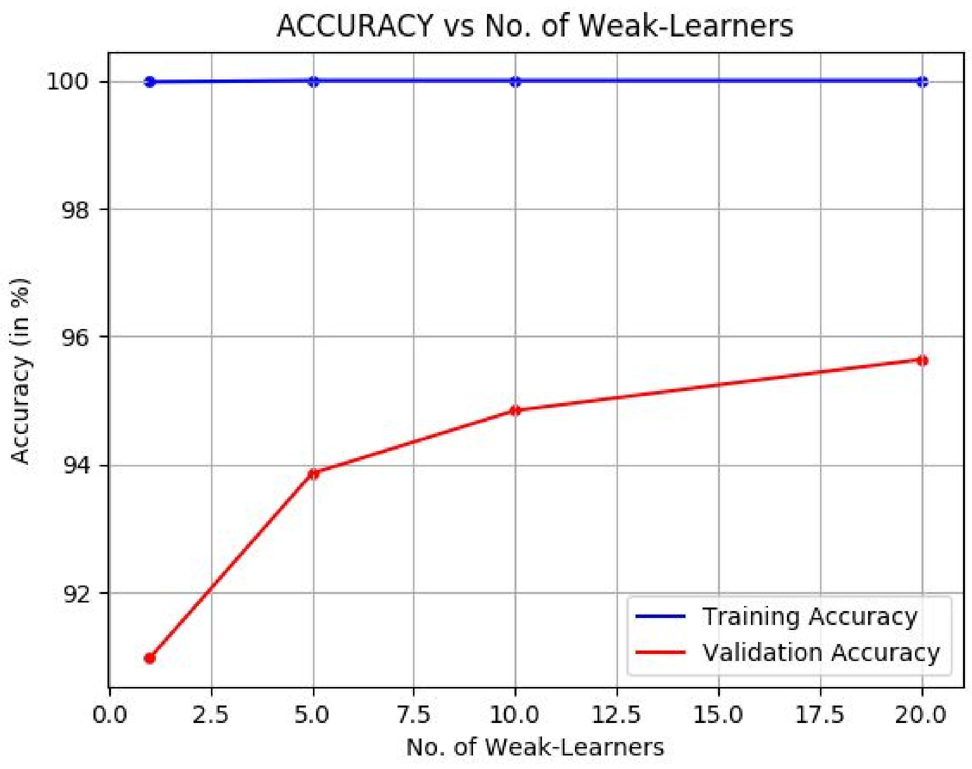
B. **Using the decision tree with parameter D implemented above, develop the AdaBoost algorithm as described in the slide with parameter L.**
Solution: The AdaBoost algorithm was implemented.

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

Solution:

Number of Learners	TRAINING ACCURACY:	VALIDATION ACCURACY:
L = 1	**99.97954173486089**	**90.97605893186004**
L = 5	**100.0**	**93.86126457949663**
L = 10	**100.0**	**94.84346224677716**
L = 20	**100.0**	**95.64149785144261**

D. Explain the behavior of AdaBoost against the parameter L.



Solution: The parameter L refers to the number of Learners in Adaboost algorithm. It can be observed that as we increased the value of parameter L, our validation accuracy increased. Since the decision tree attained 100% training accuracy for 1 learner only, the training accuracy was 100% for any value of L greater than 1 also. In Adaboost, the learners are not overfitted and we use the contribution of each learner to train our model, thus the algorithm also do not get overfitted and our validation accuracy keeps on increasing.