CS534 - Machine Learning

Implementation Assignment 3

Fall 2019

Praveen Ilango (933597560) - 33.3%
Fransiskus Derian (933296975) - 33.3%
You-Jen Lin (933663373) - 33.3%

Oregon State University
Nov 25, 2019

Introduction

In this document, our group reports the accuracy and behavior of different algorithms such as Decision Tree, Random Forest, and AdaBoost. Each part and documentation of the report is described and labeled according to the algorithm approach.

Part 1 (20 pts) : Decision Tree (DT). For this part we are interested in using a decision tree with below configuration:

- The DT uses gini-index to measure the uncertainty. Specifically if we have a node split from training list A to two left and right list AL and AR as depicted in figure 1 then

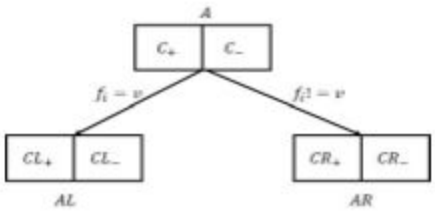


Figure 1: Split according to feature f_i testing against value v

the benefit of split for feature f_i against value v is computed as:

$$B = U(A) - p_l U(AL) - p_r U(AR) \tag{1}$$

Where U is the uncertainty measure. For this assignment, we will use gini-index, which is computed for a list such as AL as follows:

$$U(AL) = 1 - p_+^2 - p_-^2 = 1 - (\frac{CL_+}{CL_+ + CL_-})^2 - (\frac{CL_-}{CL_+ + CL_-})^2 \tag{2}$$

and p_l and p_r are the probabilities for each split which is given by

$$p_l = \frac{CL_+ + CL_-}{C_+ + C_-} \tag{3}$$

- The features are categorical with more than 2 possible values in most cases. So a feature might be tested multiple times against different values in the tree. However, when creating one-hots, we essentially make each combination of feature/value a feature itself. Note that this data does contain missing features. For this assignment, we will use a simple strategy of treating "missing" which is denoted as "?" in the data, as another possible value for the categorical feature. As mentioned earlier, this is reflected in the pre-processing (headers) that we have done.

Please implement below steps:

(a) Create a decision tree with maximum depth of 2 (root is at depth=0) on the train data using binary splits. It is of course possible to avoid doing one-hot vectors on discrete data when using decision trees and extending the impurity to handle more than 2 children at each node (and sometimes this actually performs better), but for simplicity we have provided it in a format set up for binary splits.

Answer:

This part is done in the code.

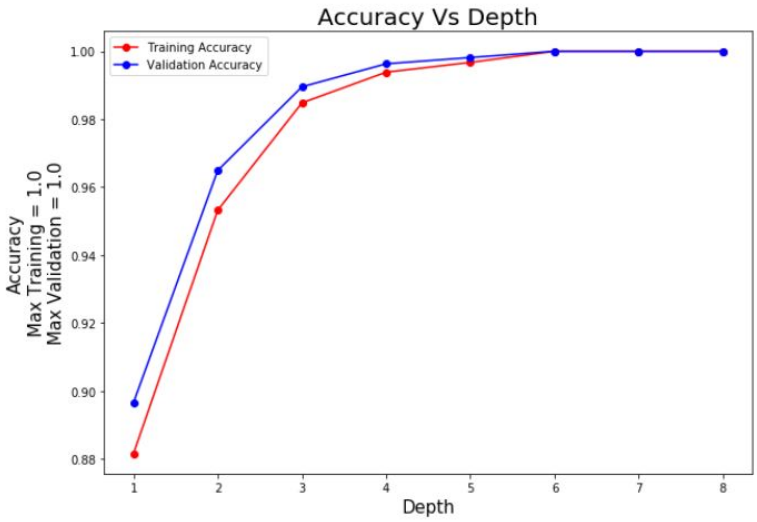
(b) Using the created decision tree, compute the train and validation accuracy. (Note: this may seem over-simplified with depth 2, but it is possible to fit this data perfectly, so we are restricting the size).

Answer:

Depth	Train Accuracy	Validation Accuracy
1	0.8816167418957734	0.8966153846153846
2	0.9532211735740664	0.9649230769230769

(c) Now, create trees with depths ranging from 1 to 8 (inclusive). Plot and explain the behavior of train/validation performance against the depth. At which depth does the train accuracy reaches to 100% accuracy? If your tree could not get to 100% before the depth of 8, keep on extending the tree in depth until it reaches 100% for the train accuracy.

Answer:



Behavior: By looking at the plot, we can see that both training and validation accuracy tend to increase as we increase the depth of the decision tree. Additionally, after training accuracy reaches 100%, the validation accuracy and training accuracy tend to be equal at 100%.

Train accuracy reaches the 100% accuracy at depth = 6.

(d) Report the depth that gives the best validation accuracy? You will probably hit a point where it will not need to be deeper, and it's best to use the simplest version.

Answer:

For this data, the best validation accuracy is reached at depth = 6, the same depth where training accuracy reaches 100%.

Best Validation Accuracy:

Depth = 6

Validation Accuracy = 100%

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 4874 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 the feature set and then pick f_i with highest benefit from m sampled features.

Answer:

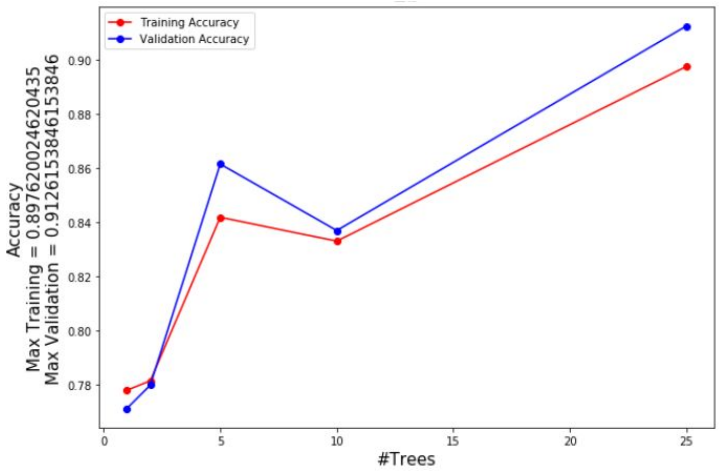
This part is done in the code.

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

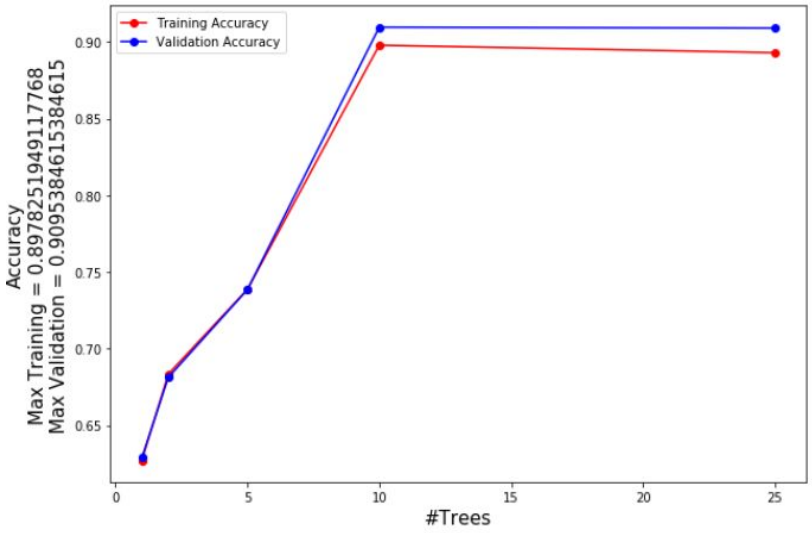
Answer:

For testing purposes, we set the random values at specific seeds:

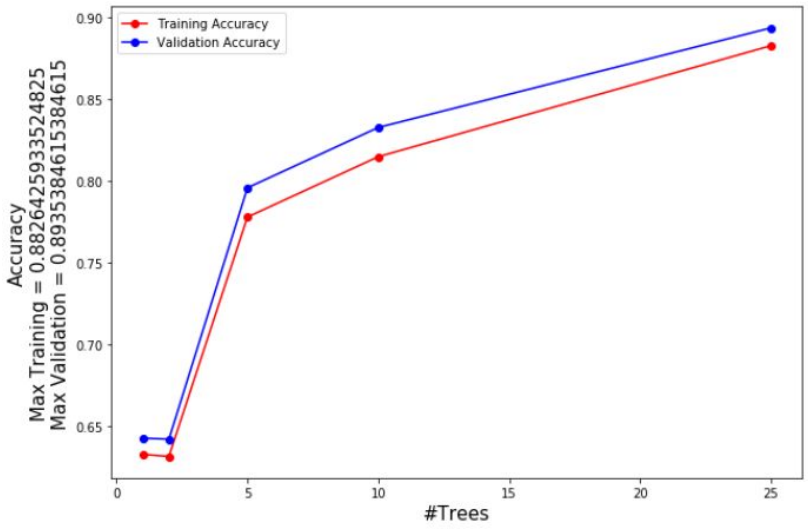
(1) Run 1: random.seed(0)



(2) Run 2: random.seed(100)



(3) Run 3: random.seed(534)



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

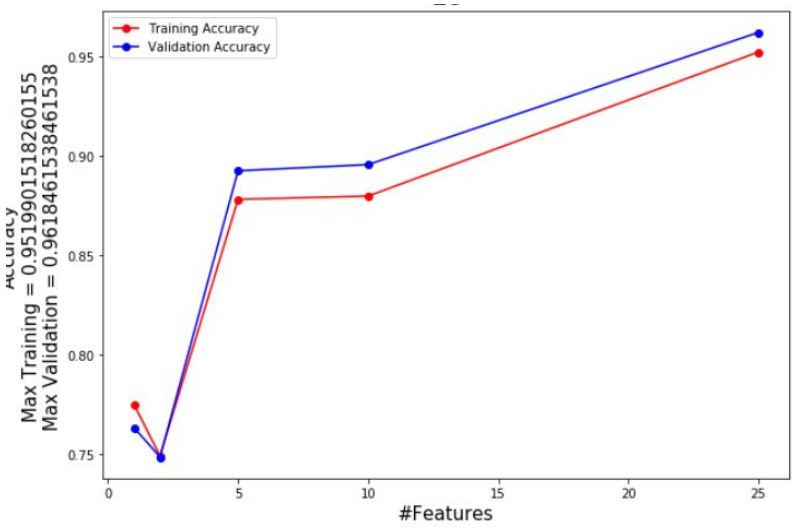
Answer:

As seen from the plots represented in part b, we can see that adding more tree into a forest tend to increase the training accuracy, while the validation accuracy seems to be increasing as well.

(d) Repeat above experiments for $d = 2$, $n = 15$ trees, and $m \in [1, 2, 5, 10, 25, 50]$. How does m change the train/validation accuracy? Why?

Answer:

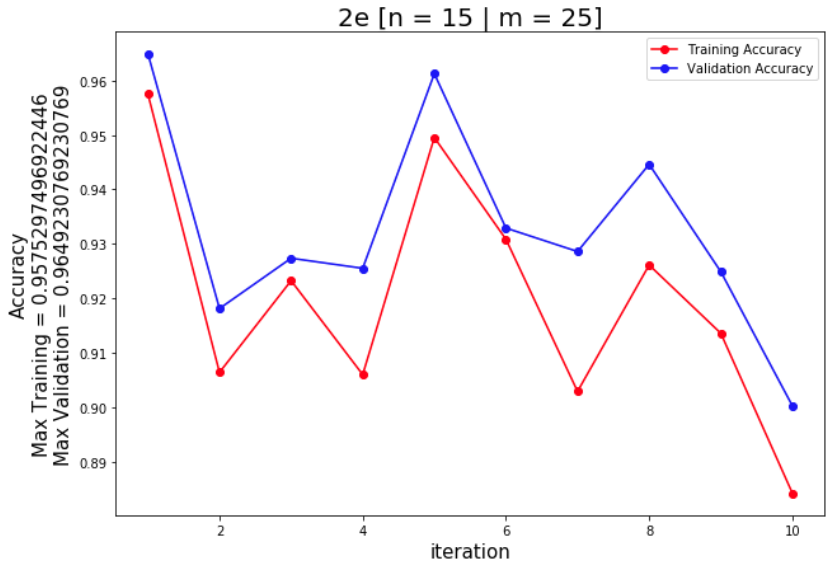
1) **random.seed(534)**



Using random.seed(534), the plot generated above shows that as we increase m, both training and validation accuracy tend to increase.
This is happening because as we increase the number of features (m), the probability of picking features which are good at differentiating the class label also increases.

(e) With your best result, run 10 trials with different random seeds (for the data/feature sampling you can use the same seed) and report the individual accuracy's, as well as the average train/validation accuracy across the 10 trials. Overall, how do you think randomness affects the performance?

Answer:



# of Run	Training Accuracy	Validation Accuracy
1	0.9575297496922446	0.9649230769230769

2	0.906442347148133	0.9181538461538462
3	0.9232663110381617	0.9273846153846154
4	0.9060320065654494	0.9255384615384615
5	0.9495281083299139	0.9612307692307692
6	0.9308576118178088	0.932923076923077
7	0.9029544521953221	0.9286153846153846
8	0.926138695116947	0.9446153846153846
9	0.9136233073450964	0.924923076923077
10	0.884283955683217	0.9003076923076923
Average	0.9200656544932295	0.9328615384615384

Min Accuracy:
Training Accuracy = 0.884283955683217
Validation Accuracy = 0.9003076923076923

Max Accuracy:
Training Accuracy = 0.9575297496922446
Validation Accuracy = 0.9649230769230769

Average Accuracy:
Training Accuracy = 0.9200656544932295
Validation Accuracy = 0.9328615384615384

Based on the data generated above, the randomness tend to fluctuate the performance / accuracy for both training and validation data. However, the accuracy for both training and validation data tend to be still relatively similar (the range between min and max values are not big).

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 only 1.** The decision tree should get a weight parameter D which is a vector of size 4874 (train size). Implement the decision tree with parameter D such that it considers D in its functionality.
(Hint: It changes the probability estimation used for computing gini-index and also for the predictions at leaves, use majority "weights" instead of majority counts).

Answer:

This part is done in the code.

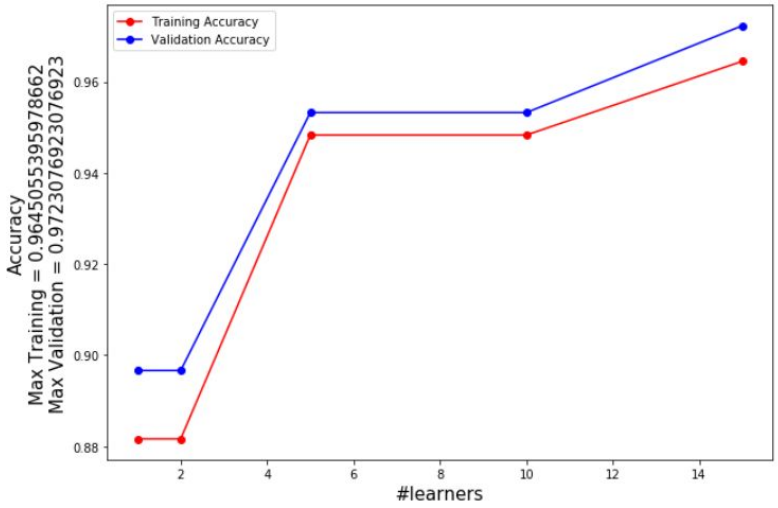
(b) Using the decision tree with parameter D implemented above, develop the AdaBoost algorithm as described in the slide with parameter L (number of base classifiers).

Answer:

This part is done in the code.

(c) Report the train and validation accuracy for $L \in [1, 2, 5, 10, 15]$.

Answer:



L	Training Accuracy	Validation Accuracy
1	0.8816167418957734	0.8966153846153846
2	0.8816167418957734	0.8966153846153846
5	0.948297086581863	0.9532307692307692
10	0.948297086581863	0.9532307692307692
15	0.9645055395978662	0.9723076923076923

(d) Explain the behavior of AdaBoost against the parameter L.

Answer:

As the number of learners increases, both the training and validation accuracy also increase. This happens because as we increase L, we are using larger number of decision trees to calculate our predictions.

(e) Repeat with depth of 2 and $L = 6$. How does this compare to the best train/validation performance with depth of 1 and $L = 15$?

Answer:

Criteria	Training Accuracy	Validation
Depth = 1, L = 15	0.9645055395978662	0.9723076923076923
Depth = 2, L = 6	0.9942552318424293	0.9956923076923077

By looking at the table above, we can see that although the number of learners is smaller (6 vs 15), the deeper the depth (2 vs 1) tends to give a better accuracy. The table shows Depth 2 and L = 6 is better than Depth = 1 and L = 15.

(f) Using your best AdaBoost, generate predictions for **pa3_test.csv**. You should include only one column, which is the class prediction for the mushroom (0 or 1). Preferably without a header or index (just ensure the index isn't shuffled - same order and length as the test csv).

Answer:

pa3_test.csv is generated and submitted along with the code.