CS4361/5361 Machine Learning

Fall 2019

Lab 3 - Decision and regression trees

Due Monday, October 7, 2019. Submit a paper copy by 5:00 p.m.

Email report to *olacfuentes@gmail.com*, include UTEP-ML2019 in the subject line.

Your task for this lab is to extend your implementation of decision and regression trees and to implement three algorithms for generating ensembles.

Part 1

Extend the decision tree functions as follows:

1. Write a function to remove unnecessary internal nodes from your decision tree. A node is unnecessary if both of its children are leaves and they result in the same classification. Perform a post-order traversal on your tree, replacing each unnecessary internal node by the corresponding classification. To test your implementation, make sure the accuracy before and after pruning is the same.
2. Write functions to determine the number of internal nodes in your decision and regression trees.
3. Write functions to determine the height of your decision and regression trees.
4. Write a function to determine the importance of each attribute in the dataset from the structure of the tree. A good measure of an attribute's importance is the average number of times it is used to predict the target function value of each training example.
5. (Extra credit) Optimize your prediction time by eliminating the loop in the predict function. This will require to modify the classify function as well.

Part 2

Implement the following ensembles of predictors and evaluate their performance:

1. Randomization
2. Bagging
3. Boosting

Write a report including (at least) the following items:

1. Problem description
2. Algorithms implemented
3. Experimental results. It is important that you try to obtain the best possible results by optimizing the parameters of your trees and ensembles. Show accuracies or mean squared errors and running times for each algorithm and parameter choice. You may want to use tables and or plots to illustrate this.
4. Discussion of results. What ensemble and parameter selection yields the best results? Why?
5. Conclusions
6. Appendix: Source code

Appendix

Sample run:

```
Original Tree:
                 class= 1
             if x[0] <= 43.3278
                 class= 1
         if x[6] <= 24.0087
                 class= 1
             if x[3] <= 0.238304
                 class= 1
     if x[8] <= 9.49499
                 class= 1
             if x[1] <= 14.6079
                 class= 1
         if x[0] <= 45.6028
                 class= 1
             if x[0] <= 86.9231
                 class= 0
 if x[8] <= 27.6336
                 class= 1
             if x[2] <= 2.30643
                 class= 1
         if x[2] <= 2.50051
                 class= 0
             if x[1] <= 16.3897
                 class= 1
     if x[0] <= 53.3287
                 class= 0
             if x[9] <= 198.209
                 class= 0
         if x[1] <= 49.4408
                 class= 0
             if x[5] <= -63.1967
                 class= 0
Number of internal nodes: 15
Height: 4
train accuracy: 0.80744
test accuracy: 0.81519

Pruned Tree:
         class= 1
     if x[8] <= 9.49499
             class= 1
         if x[0] <= 45.6028
                 class= 1
             if x[0] <= 86.9231
                 class= 0
 if x[8] <= 27.6336
             class= 1
         if x[2] <= 2.50051
                 class= 0
             if x[1] <= 16.3897
                 class= 1
     if x[0] <= 53.3287
         class= 0
```

```
Number of internal nodes: 7
Height: 4
Attribute usage per example:
Attribute: 0 used 0.7169 times per training example
Attribute: 1 used 0.1316 times per training example
Attribute: 2 used 0.2780 times per training example
Attribute: 3 used 0.0000 times per training example
Attribute: 4 used 0.0000 times per training example
Attribute: 5 used 0.0000 times per training example
Attribute: 6 used 0.0000 times per training example
Attribute: 7 used 0.0000 times per training example
Attribute: 8 used 1.6120 times per training example
Attribute: 9 used 0.0000 times per training example
Most important attribute: 8
train accuracy: 0.80744
test accuracy: 0.81519
```