COM307 Project 3 Write-up: Ionosphere Free Electrons

Khe (Kate) Le

Due date: Nov 2

1. My data

- I used *ionosphere.data.csv* from UCI with 351 instances and 34 variables
- I applied *sampling with replacement* to the original dataset, so the same instance/row may be chosen and added to the sample more than once. This is because:
    - The more instances I add, the less likely I am to capture a sufficiently large number of all possible combinations of all instances to accurately reflect the ionosphere's types ('good' or 'bad')
    - Classification model is vulnerable because it breaks data up into smaller subsets, and builds trees which tend to overfit to the sample data and fail on real life data. With replacement, the random forest algorithm can avoid building a model specific to the dataset thus reducing this overfitting.

2. How I selected m random variables from amongst 34 variables

- Since only a fixed subset of m variables are considered at each point a split is made, I chose m to be the square root of the number of input variables. Given 34 input variables, $m = \sqrt{34} \; = \; 5$ (rounded down). This is because:
    - According to the authors of the original paper Random Forest, the number of randomly selected variables can influence the generalization error in 2 ways:

        - Selecting many variables increases the strength of the individual trees
        - Reducing the number of variables leads to a lower correlation among the trees thus increasing the strength of the random forest.
    - It is also recommended to choose $\sqrt{N}$ or $\log N$ variables from among N variables based on empirical data. It has shown that lower correlation among trees can decrease generalization error enough to more than offset the decrease in strength of individual trees.

3. **How I figured which variable does the best job separating 'good' from 'bad' and at what value (= what I did when I reached decision point)**

- To find the best split point, I evaluated the cost of each value in the training dataset for each input variable* using the Gini index, because it helps calculate the amount of probability of a specific variable that is classified incorrectly when selected randomly.
- In the case of a two-class classification problem like this, a Gini index of 0 is perfect purity where class values ('good' or 'bad') are perfectly separated into two groups.
- My goal is to find a split with the lowest cost or a value x such that the two groups splitted by this value have the lowest Gini index.
- An example:
    - Consider a set of 5 random variables at indices 0, 2, 4, 6, and 8
    - Consider the variable at index 0
    - Consider the first value '0.02' of this variable (at 1st row, 1st column of *ionosphere.data.csv)*

- Traverse through all values in the remaining instances/rows of this variable. If the value in consideration is < 0.02, append that value to the list ‘left’. Else if it is >= 0.02, append it to the list ‘right’.
- Calculate the Gini index i of the group of the two lists ‘left’ and ‘right’
- If i < minimum Gini index, save i as the new minimum Gini index.
- Repeat the procedure for other remaining values of the variable and other variables
- Eventually, we’ll find a value x (among 351 instances) of a variable y (among 5 variables) that yields the lowest Gini index. Use it as the split point.

**Instead of considering all input variables, I only considered a smaller amount of the total variables (explained in 2 where I took the square root). For example, I will only consider 5 variables out of 34 variables each time I need to make a split. These 5 variables will be chosen randomly and without replacement.*

4. How I merged the trees I made into a random forest

- I made predictions with a single tree by navigating the tree with the specifically provided row of data.
- I implemented this with a recursive function, calling the same prediction steps again with the left or right child nodes depending on how the split affects the provided data. The function then checks if a child node is either a terminal value to be returned as the prediction, or if it is a dictionary node containing another level of the tree to be considered.
- After making predictions with all the trees, I made the predictions of the random forest by averaging the predictions of each individual tree (bagging)