Supervised Learning: Ensemble Methods

April 16, 2021

1 Motivation

When building models in previous sections, getting a better model often means you sacrifice the simplicity of the calculations of that model. How do we take simple models that are easy to compute, and combine them to maximize their strengths and minimize their weaknesses?

2 Two Methods

- Bagging: Average multiple predictions to get one aggregate conclusion
- Boosting: Combing the strengths of each method and getting a best answer by utilizing strengths and mitigating violence

3 Problems with Combining Models

Combining methods yields some problems, namely the competing variables bias and variance.

- Bias: assume too many things and get a simpler model. High bias typically means we do not fit well to the test data.
- Variance: between each point, the predictor function changes direction a lot. High variance does fit well to training data but is unable to generalize well

4 Dealing with the Bias-Variance Problem

We can do this by building models that do better when we ensemble them to meet in the middle in terms of variance and bias. We can do this by introducing randomness by bootstrapping the data ro use only a subset of features when doing each split of the tree or step of the algorithm.

4.1 Example: Random Forests

When constructing a decision tree, choose a random set of features and create a decision tree with that. Do this multiple times and get multiple trees. When a new point comes in, put the point through each tree and then use each tree's classification decision as a vote, and vote for the outcome.

5 Bagging Algorithm

The algorithm for a 2 variable system goes as following:
Take the data, plot it, and fit a horizontal or vertical line to it, and then do it a bunch of times. Superimpose each result over one another and for each new point, treat each horizontal and/or vertical line fitted and have them vote for which place to classify the point to.

5.1 Ada Boost

Similar, but it punishes the misclassified points more in the next iteration.
First, assign each point a value of 1. Fit a line to it. After fitting the line as best as possible, count up the points for correctly and incorrectly classified data. Then, for each data point that was incorrectly classified, scale the point value assigned to the data such that when you sum up values of the incorrect points, you get the same total point value as the correct data. Use these point values for the next iteration. When evaluating new data, we do the same thing as before, figure out what each model "votes" on, and make our prediction the most voted. But this time, we weigh a model's vote more if it is more accurate. We calculate the accuracy A of each model, and weigh each model's vote $Weight = ln(\frac{A}{1-A})$