P556 Homework 4

Fall 2022

Due on Dec 9th, 11:59pm

Question 1: Decision Trees (40 points)

For the first two problems, it would be helpful for you to draw the decision boundary of your learned tree in the figure.

- Consider the problem of predicting if a person has a college degree based on age and salary. The table and graph below contain training data for 10 individuals. Now build a decision tree for classifying

Age	Salary ($)	College Degree
24	40,000	Yes
53	52,000	No
23	25,000	No
25	77,000	Yes
32	48,000	Yes
52	110,000	Yes
22	38,000	Yes
43	44,000	No
52	27,000	No
48	65,000	Yes

Figure 1: Question 1

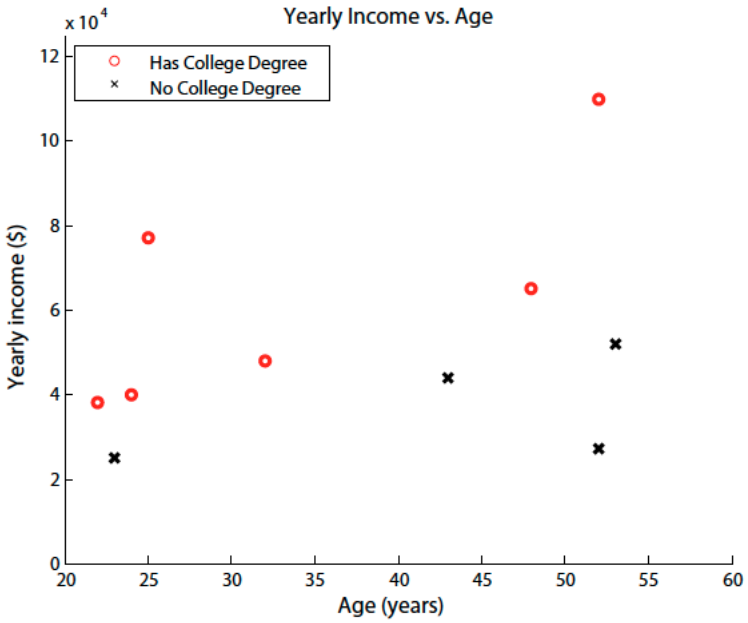

Figure 2: Question 1

whether a person has a college degree by greedily choosing threshold splits that maximize information gain. What is the depth of your tree and the information gain at each split?

- A multivariate decision tree is a generalization of univariate decision trees, where more than one attribute can be used in the decision rule for each split. That is, splits need not be orthogonal to a feature's axis.

  For the same data, learn a multivariate decision tree where each decision rule is a linear classifier that makes decisions based on the sign of $\alpha x_{age} + \beta x_{income} - 1$. Draw your tree, including the α, β and the information gain for each split.

- Multivariate decision tres have practical advantages and disadvantage. List advantages and disadvantages you can think of when comparing multivariate decision trees to univariate decision trees.

Question 2: Bagging and Boosting (50 points)

In this question, you will need to implement the boosting algorithm AdaBoost as well as a Bagging, in both cases using decision trees as the base classifiers. You will then be asked to experiment with your implementation on a number of provided datasets, and to write up a brief report on your findings. (Minor note on terminology: Classifier means the same thing as hypothesis, base classifier, and weak classifier in the context of boosting.)

- The first part is that you need to write a function to implement the AdaBoost algorithm we discussed in class. The input of your function is the training and test sets, as well as the number of rounds of boosting T. It should then run AdaBoost for T rounds, using the decision-tree algorithm as the base learner. The function should then return the predictions of the final combined classifier on the given training and test examples, as well as the training and test error rate of the combined classifier following each of the T rounds.

- "Bagging" (short for "bootstrap aggregating") is a different method for combining decision trees or other base classifiers. Similar to boosting, the base learning algorithm is run repeatedly in a series of rounds. However, the manner in which the base learner is called is different than in boosting. In particular, on each round, the base learner is trained on what is often called a "bootstrap replicate" of the original training set. Suppose the training set consists of m examples. Then a bootstrap replicate is a new training set that also consists of m examples, and which is formed by repeatedly selecting uniformly at random and with replacement m examples from the original training set. This means that the same example may appear multiple times in the bootstrap replicate, or it may appear not at all. Thus, on each of T rounds of bagging, a bootstrap replicate is created from the original training set. A base classifier is then trained on this replicate, and the process continues. After T rounds, a final combined classifier is formed which simply predicts with the majority vote of all of the base classifiers.

We will use three real world datasets:

- The first dataset is a **letter** dataset contains descriptions of the characters "C" and "G", and the goal is to distinguish between these two letters. The class label is either "C" or "G". There are 16 attributes for things like the width of the letter and the total number of pixels turned on. There are 500 training and 1009 test examples. More detailed information about this dataset and the various attributes is available here (obviously, we used only the letters C and G). The dataset is available at https://archive.ics.uci.edu/ml/datasets/letter+recognition

- The **credit** dataset classifies people described by a set of attributes as good or bad credit risks. There are 20 attributes encoding credit history, purpose of the loan, employment status, etc. There are 400 training and 600 test examples. More detailed information about this dataset and the various attributes is available here (we used the "original" dataset, and did not make use of the "cost matrix"). The dataset is available at https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

- The **spam** dataset classifies email messages as spam or ham. The 57 attributes mainly encode the number of times that certain words or characters occur. There are 1000 training and 3601 test examples. The dataset is available at https://archive.ics.uci.edu/ml/datasets/spambase

Making sure that algorithms of this kind are working correctly is essential, but can be difficult since it is not always obvious what correct behavior looks like. For this reason, it is important to take extra care to test your implementations. One way of doing so is by trying small datasets on which it might be possible to compute the correct answer by hand.

Once you have your boosting and bagging implementations working, your next step is to run some experiments using them on the provided datasets. You should try out both boosting and bagging on the three provided datasets. You also should experiment both with deep trees and very shallow trees (say, of depth only one, often called "decision stumps"). Thus, there are (at least) twelve combinations to consider: there are three datasets, on each of which you can try any combination of bagging or boosting on deep or shallow trees.

Your aim should be to compare the algorithms, and to try to figure out which algorithm is most effective when, and why.

The end result of these experiments should be: 1) your code; 2) a brief written part, describing in precise terms what experiments you ran, what observations you made, and what you can say about the questions and issues discussed above. You certainly should include plots showing the results of your experiments. This report should include at least the following:

- plots of test error for the various algorithms and datasets as a function of the number of rounds;
- a discussion of how the algorithms compare in performance and behavior;
- other observations, experiments, plots, etc. that are the result of further exploration.