Homework 3

Due: Oct 31, 2022 at 11:59pm EST

Instructions: This homework requires answering some open-ended questions, short proofs, and programming. This is an individual assignment, not group work. Though you may discuss the problems with your classmates, you must solve the problems and write the solutions independently. As stated in the syllabus, copying code from a classmate or the internet (even with minor changes) constitutes plagiarism. You are required to submit your answers in pdf form (use LaTeX) in a file called `hw3.pdf` to Gradescope under "HW3". Code should also be submitted to Gradescope in a file called `hw3.py` under "HW3 Programming". A LaTeX template for the pdf submission and a code skeleton for the code submission are available on Piazza. Please do **not** modify the base code in the coding skeleton; simply augment it with your solution. Late submissions do not receive full credit, except in extenuating circumstances such as medical or family emergency. Submissions submitted 0-24 hours late are eligible for only 90%, 24-48 hours late for 80%, 48-72 hours late for 70%, and later than 72 hours for 0% of the total credit for this assignment. Late days may be used (if available) to avoid these penalties.

The total assignment is worth 75 points.

Problem 1 (15 points)

We have studied one definition of expressiveness of machine learning models in terms of weak vs. strong learners; a related concept is the notion of parametric vs. non-parametric models. A machine learning model is said to be *parametric* if the complexity of the model (e.g., the number of parameters) does not grow as a function of sample size (i.e., the number of rows of data.) A model is said to be *non-parametric* if the complexity of the model does grow as a function of sample size.

(i) Provide a formal justification (by referencing the model complexity given training datasets of increasing size) for each of the following claims:

(a) Linear logistic regression is parametric. (2.5 points)
(b) Decision trees can be parametric or non-parametric. (5 points)
(c) Random forests are non-parametric. (2.5 points)

(ii) Discuss the bias-variance tradeoff of decision trees as a function of depth. How does this justify bagging as a strategy for mitigating expected test error in non-parametric decision trees? (5 points)

Problem 2 (15 points, 5 points per question)

In this question we discuss the big O computational complexity of finding the best feature and threshold for splitting the data at a given step of the decision tree algorithm.

Recall that the best split can be found by iterating over all d features and all unique values of these d features in the dataset, and comparing the entropy of splitting on these values: one split gets all rows of data where the feature is less than the value; the other split gets all rows of data where the feature is greater than or equal to the value. Answer the following questions. For convenience, assume that the outcome is binary, and that all the features are continuous with no repeated values of the features in the dataset.

(i) What is the computational complexity of an algorithm for computing the entropy of a dataset with n rows and d features?

(ii) What is the computational complexity of a naive algorithm for finding the best split of data that computes the entropy from scratch for each possible split?

(iii) Let's say the values for each feature were being iterated over according to a pre-sorted order. How can we improve the computational complexity described in (ii) by caching computations of the entropy for previous values of the feature?

Problem 3 (15 points)

The following are short conceptual questions related to bagging and boosting.

(i) How does the random forest algorithm differ from bagging? (2.5 points)

(ii) Can we apply an embarassingly parallel strategy to speed up the training process for bagging and/or boosting? Briefly justify your answer. (5 points)

(iii) Describe why the number of bootstraps m is not a hyperparameter that requires much tuning in the case of bagging, however, the number of iterations m in boosting does require tuning. (5 points)

(iv) Is it possible to bag/boost linear models? What are the limitations? (2.5 points)

Problem 4 (30 points)

In this problem you will implement and apply decision trees based on the code skeleton provided in `hw3.py` and helper functions provided in `data_loader.py`. There are two datasets provided to help test your implementation: an artificially generated dataset in `simulated_data.csv` with features X_1, X_2 and outcome Y, and a bone marrow transplant dataset from the Wroclaw Medical University, Poland[1]. The features in the bone marrow transplant data include information about the bone marrow donor and recipient (the recipients are all children/young adults between the ages of 0 and 20.). The outcome of interest is whether the transplant is accepted by the recipient's body, and leads to long-term survival.

The helper functions in data loader are meant to assist with your implementation and analysis. Note the implementation of decision trees is more involved, so it has been broken down into some extra steps. It's advisable to follow these in order.

(i) Implement the `predict` method using depth first traversal of the tree. (6 points)

(ii) Implement the `weighted_entropy` and `_get_best_split` method using the naive algorithm described in Problem 2 (since we do not assume a pre-sorted order.) (6 points)

(iii) Implement a **recursive** `_build_tree` method that is used in the `fit` method. The base

[1]Original data from Kalwak et al (2010); data processed by Sikora et al (2020)

cases for stopping the recursion, where it will return a leaf node, are that either (i) all values of the outcome in the provided data are the same (all 1s or all 0s), or (ii) the current depth is greater than or equal to the max depth. Otherwise, the method creates a new vertex, recursively builds a left subtree and right subtree for this vertex based on the best split, and returns the vertex. (8 points)

(iv) Typically, we define training, validation, and test sets to fit a model, find an optimal setting of the hyperparameters, and compute the generalization error respectively. However, the number of examples in the transplant data is fairly low, so after we produce a train and test split, we bootstrap 5 training and validation splits to find the optimal hyperparameter (depth of the tree)

(a) What is the optimal depth found for the bone marrow transplant data? (1 point)

(b) What is the accuracy of the final model on the test set? (1 points)

(c) The authors in Kalwak et al (2010) state that higher CD34 (white blood) cell doses in the graft yield better survival outcomes – is this reflected in your final decision tree? The coding of the 0/1 labels and meaning of each variable can be found in the `transplant.csv` file. Feel free to run the code multiple times (remember different trees may give the same results.) (3 points)

(v) Describe some strategies for improving the performance of the model in terms of reducing the bias-variance-noise components of the expected test error. (5 points)

Important submission note:
Report all answers to the above questions in the PDF that you turn in. Hand in your Python code separately on Gradescope under HW 3 Programming. Please make sure your variables and functions are aptly named, and your code is clear and readable, with comments where appropriate.