Homework 4 (due 2/23)

DS-210 @ Boston University

Spring 2022

Before you start…

Collaboration policy: You may verbally collaborate on required homework problems. However, you must write your solutions independently without showing them to other students. If you choose to collaborate on a problem, you are allowed to discuss it with at most 2 other students currently enrolled in the class.

The header of each assignment you submit must include the field "Collaborators:" with the names of the students with whom you have had discussions concerning your solutions. If you didn't collaborate with anyone, write "Collaborators: none.“ A failure to list collaborators may result in a credit deduction.

You may use external resources such as software documentation, textbooks, lecture notes, and videos to supplement your general understanding of the course topics. You may use references such as books and online resources for well known facts. However, you must always cite the source.

You may **not** look up answers to a homework assignment in the published literature or on the web. You may **not** share written work with anyone else.

Submitting: Solutions should be submitted via Gradescope. The entry code is 3Y85PZ. Please submit a solution to this homework as a single IPython notebook (`.ipynb`) as much as this is possible.

Grading: Whenever we ask for a solution, you may receive partial credit if your solution is not sufficiently efficient or close to optimal. For instance, if we ask you to solve a specific problem that has a polynomial–time algorithm that is easy to implement, but the solution you provide is exponentially slower, you are likely to receive partial credit.

Late submission policy: No extensions, except for extraordinary circumstances. We accept submissions submitted up to one day late, but we may deduct 10% of points.

Questions

1. **(4 points)** It is important—especially for the second part of this course—that you know how to edit text files. Almost every operating system comes with a text editor. You may prefer to install, however, an editor that supports programming. Useful features in such an editor may include syntax highlighting and autocompletion.

   To prove that you have a working text editor, use it to create a text file `test.txt` that contains the following text:

```
[x] I typed this text in my text editor and copied it here.
```

Next, copy and paste this text to the notebook that serves as your solution to this homework.

2. **(12 points)** Install Rust, which we will use in the second part of the course. You can do it by following instructions at https://www.rust-lang.org/tools/install. (There may exist alternate methods for your operating system. You can find a discussion of some them in the official documentation.)

   After you are done installing Rust, you should be able to run the following commands from the command line: `rustc` and `cargo`. In particular, run both `rustc --version` and `cargo --version` and include their output in your homework solution to prove that you have successfully installed Rust and to receive credit for this part of the homework.

   Note: https://doc.rust-lang.org/book/ch01-01-installation.html also discusses various options for installing Rust.

3. **(12 points)** In this problem, you are asked to use decision trees for regression with two different target loss functions: minimum square error and minimum absolute error. Design a function $f : [0, 1] \to [0, 1]$ such that when you sample input points x_i from $[0, 1]$ and use the sequence $(x_i, f(x_i))$ as your input, you are likely to see the difference between the two loss functions with **six** leafs (i.e., with `max_leaf_nodes=6`).

   Among other things, your solution should contain:

   - a plot of f and the two functions resulting from the training process under different loss functions,
   - an explanation of the differences and how they are a result of differences between the two loss functions.

   Note: Make sure you read the documentation for `sklearn.tree.DecisionTreeRegressor`.

4. **(12 points)** Recall that `numpy.polyfit` can be used to find a bounded degree polynomial that fits data well. Show clear examples of overfitting and underfitting occurring when learning a function f from $[0, 1]$ to $\mathbb{R}$. Support your examples with error estimates. Your examples can feature noisy data points.