Midterm: Regression Trees

Purposes

- Understanding regression trees.
- Studying the Classification And Regression Tree (CART) algorithm to build a binary regression tree for a given training dataset.
- Implementing a basic CART algorithm to build a binary regression tree of a bounded height.
- Approximation of continuous functions using regression trees.
- Approximation of dynamical systems using regression trees.

Implementation Tasks

1. (10 points) Read the attached notes from Dr. Cosma Shalizi to study the basic algorithm for building a regression tree. Here, we consider a simplified version which does not prune the tree. Instead, we let the user to specify a maximum height (depth) or a leaf size to terminate the building process of a tree. Therefore, we focus on the two questions: *(1) how to find the best feature to split the dataset,* and *(2) how to choose a proper split value.* Please read and study the lecture notes. After the study, you are required to implement a Python class named `RegressionTree` which must have the following components:

   - (4 points) A constructor which takes a given set of training data and build a binary regression tree. The function must have at least the following parameters: the dataset, the maximum height of the tree, the leaf size, and a variable which indicates which limit (height or leaf size) is used to control the complexity of the tree. The two limits may also be specified as `None` to let the tree to span as much as possible, i.e., every node has exactly one sample.
   - (3 points) A function named `predict` which returns the prediction for a given input sample.
   - (3 points) A function named `decision_path` which displays the decision path of a given input sample, that is the sequence of deduction rules to obtain the prediction.

   In the notes, the "best" feature for a split along with the split value are obtained by measuring the drop of sum of squared errors which is very similar to the strategy used by scikit-learn (Section 1.10.7).

2. (6 points) Test your implementation using the continuous function $y = 0.8\sin(x-1)$ over the domain $x \in [-3, 3]$. You may generate ≈ 100 training samples which are uniformly distributed in the domain and compute their labels, i.e., y values. Randomly make an 80% to 20% split for training and test data. Please test your implementation in the following way:

   - (2 points) Do not apply any limitation, take down the test result: the height of the resulting regression tree, the test error, and the time cost of building the tree.
   - (2 points) Apply a limit to the tree height. You may use 1/2 and 3/4 of the height obtained without any limit. Take down the test result.
   - (2 points) Apply a limit to the leaf size, that is, a leaf is allowed to have a limited number of training samples. You may consider the size of 2, 4 and 8. Take down the test result.

   Create a table to compare the test results you collected, to better understand the impact of height limit and leaf size limit on the overall prediction accuracy.

3. (3 points) Let us consider a multiple dimensional case. A discrete-time dynamical system can be defined by a difference equation $\bar{x}_{k+1} = f(\bar{x}_k)$ wherein the vector $\bar{x}_i$ denotes the system state at the time $t = i$. For example, the move of a vehicle for every 0.1 seconds can be described by $x_{k+1} = x_k + 0.1v_k$, $v_{k+1} = 10$ such that x denotes the location of the vehicle and v is the velocity which is always $10m/s$.

In this task, you are required to design a method using regression tree to approximate multidimensional functions. In the above vehicle example, the system has a state which is 2-dimensional (x, v). If we do not know the exact system model (equations) but just have a set of paired data of the form $((x_k, v_k), (x_{k+1}, v_{k+1}))$, how would you like to use a regression tree-based model to predict the next state of the system? Please explain your solution.

4. (6 points) You are required to evaluate your solution based on the following case studies and take down the experimental results.

   - (2 points) System model: $x^{(1)}_{k+1} = 0.9x^{(1)}_k - 0.2x^{(2)}_k$, $x^{(2)}_{k+1} = 0.2x^{(1)}_k + 0.9x^{(2)}_k$. The system has two state variables $x^{(1)}, x^{(2)}$. You need to write a function to generate a set of samples of the form $((x^{(1)}, x^{(2)}), (x^{(1)\prime}, x^{(2)\prime}))$ such that $(x^{(1)\prime}, x^{(2)\prime})$ is the next state of $(x^{(1)}, x^{(2)})$. You may only consider the range of $x^{(1)} \in [-5, 5]$, $x^{(2)} \in [-5, 5]$.

     To better visualize the approximation quality, you may additionally create figures showing the evolutions (trajectories) of $x^{(1)}, x^{(2)}$ along with time. You may choose the initial state $x^{(1)}_0 = 0.5$, $x^{(2)}_0 = 1.5$, predict the future states for $t = 1, 2, \ldots, 20$ and comparing the predictive trajectories and the actual trajectories.

   - (3 points) We consider the following program:

```
def func(x):
    z = 0
    for _ in range(20):
        if x > 1:
            x = 0
        else:
            x = x + 0.2
        z = z + x
```

     We consider the values of the variables `x` and `z` at the beginning of each iteration as the state of the program/system. We only consider the range: $x \in [-3, 3]$, $z \in [0, 15]$. You are required to build a regression tree-based model to predict the next state for the program. Similar to the previous example, you may choose the initial state $x_0 = 2$, $z_0 = 0$, predict the future states for $t = 1, 2, \ldots, 20$ and comparing the predictive trajectories and the actual trajectories.

   - (1 point) Tune the hyperparameters (tree heights, leaf sizes) for the models and try to obtain best fits. Show the test errors and time costs of building the regression models.

   You may use the NumPy function `numpy.random.uniform` to randomly select a set of samples and use the function `sklearn.model_selection.train_test_split` to subdivide the training and test data. We may consider the ratio of 80% (training) to 20% (test).

Report

You are required to write a report for this assignment. It should include the following parts:

- For Task 1, please explain tree data structure, the selection of the split dimension and value, and give the pseudo code of tree construction.
- For Task 2, please give the experimental results and their explanations.
- For Task 3, please give the explanation of your method.
- For Task 4, please give the experimental results and their explanations.

Please feel free to add more observations and thoughts.