CSCI 4160/6963, ECSE 4965/6965
Reinforcement Learning
Homework 3

Overview

In this homework, you will implement the decision tree algorithm. You will use the tree to classify MNIST digits, so the dataset will be the same as in the first assignment. Unlike HW2, this time you will perform the full 10-label classification task.

Since training the tree on the raw pixel values is computationally expensive and unlikely to generalize, your features will be the weights you learn in HW2. Specifically, denote by $\boldsymbol{w}_{ij}$ the weights you learn when performing regression for the pair (i, j). Then, given a new example $\boldsymbol{x}$, the corresponding feature generated by pair (i, j) will be $\boldsymbol{w}_{ij}^T\boldsymbol{x}$. Thus, you will have 45 features in total for every example, namely $[\boldsymbol{w}_{01}^T\boldsymbol{x}, \boldsymbol{w}_{02}^T\boldsymbol{x}, \dots, \boldsymbol{w}_{89}^T\boldsymbol{x}]$. As you saw in HW2, most pairs can be separated almost perfectly with linear regression, so these features should be sufficient to achieve high accuracy on the full 10-label classification task.

As discussed in class, the decision tree training algorithm consists of the following steps: for each leaf, you 1) iterate through all dimensions, 2) for each dimension, you select the optimal threshold that minimizes the 0-1 loss, 3) split on the feature that results in the lowest 0-1 loss. You only need to implement the vanilla algorithm (i.e., you do not need to add any improvements such as pruning, bagging or random forests).

I am also providing my weights from HW2 in case you would like to use those. They are in the `weights.txt` file, and I have also given you a short function to load them into numpy vectors. For reference, the weights are computed for regressions in the following order: (0,1), (0,2), …, (1,2), (1,3), …, (8,9). In any case, the weight order shouldn't matter for this homework – as far as the decision tree is concerned, it is just given 45 features to split on.

Logistics

All coding assignments in this course will be in Python. If you need help with Python, please talk to me ahead of time, so we can discuss the best way to get familiar with it.

For this assignment, you will be using your own computer, which should be sufficient. The code should not require more than 15 minutes to run on any standard laptop.

You are provided with skeleton starting code. The MNIST dataset is the same as in HW2. Please use the Python libraries in the `requirements.txt` file that's provided on LMS.

Grading

For a full score, you need to achieve 90% (**91% for graduate students**) test accuracy on the full 10-label task. You will lose 5 points for every percentage point below the target test accuracy. Your code must follow the provided skeleton code structure, in particular the `train`, `predict` and `test` functions in the `DecisionTree` class.

Make sure your code prints out some indication of progress as well as all required information – we should not have to insert any print statements in order to find out whether your code works!

Finally, please produce the plot in 1) below and provide brief answers (2-3 sentences each) to each of the questions below:

1) What happens if you increase the number of leaves/regions in your tree? Plot a graph with two curves: train and test accuracy. On the x-axis you have the number of leaves, and on the y-axis you have accuracy. You need to show enough points so that you illustrate when overfitting begins. For example, you can try the following 10 points for number of leaves: {100, 200, …, 1000}.

2) How does your threshold step size affect accuracy and computation? What happens if you try a very big step size? How about a very small step size?

Hints and Tips

- If you end up using your weights from HW2, you should store your weights in a file (e.g., a CSV file), so that you don't re-run the linear regression every time you train the tree. This way, you only need to load the weights before you train the tree.
- Start early. Implementing the tree will probably require at least 250-300 lines of code, and debugging may be tricky in some cases.
- You might want to have a Node subclass to handle the graph structure.
- You will need to choose a desired number of leaves for your tree. Make sure this is stored in a variable so that it is clear. Same goes for all other hyper-parameters (e.g., the step size if you decide to do linear search for your threshold).

Submission

Please use `LMS` to submit a zip file containing 1) your **.py** code, along with instructions on how to run it, 2) a **.pdf** file containing your answers to the above questions, 3) a **.png** file with your

accuracy plot [including the plot in the pdf file is fine also]; 4) a **.csv** file with your HW2 weights so we can run your code (if you used my weights, please submit those instead). The deadline is **11:59pm, Thursday, Oct. 2**.