CS 441

HW 2: Trees and MLPs

Due Date: Feb 27, 2023 11:59pm

In this assignment, we will add a penguin classification task to our repertoire and explore trees, ensemble, and MLPs. The aims of this homework are:

1. Explore the impact of model complexity while applying regression trees, random forests, and boosted regression trees.
2. Get experience with training and applying MLPs
3. Gain exposure to using correct methodologies for learning rate selection and testing
4. Get practice analyzing features and selecting and designing models for open problems

Read this document and the report template and tips and tricks before beginning to code.

- Report template
- Tips and Tricks
- Starter code

1. **Model Complexity with Tree Regressors** [30 pts]

One measure of a tree's complexity is the maximum tree depth. Train tree, random forest, and boosted tree regressors on temperature regression (data), using all default parameters except:

- `max_depth={2,4,8,16,32}`
- `random_state=0`
- For random forest: `max_features=1/3`

Measure train and val RMSE for each and plot them all on the same plot using the provided `plot_depth_error` function. You should have six lines (train/val for each model type), each with 5 data points (one for each max depth value). Include the plot and answer the analysis questions in the report.

2. **MLPs with MNIST** [40 pts]

For this part, you will want to use a GPU to improve runtime. Google Colab provides limited free GPU acceleration to all users. See Tips for detailed guidance on this problem.

First, use **PyTorch** to implement a Multilayer Perceptron network with one hidden layer with ReLU activation. The layer size has been given to you in the starter code. Set the network to minimize cross-entropy loss, which is the negative log probability of the training labels given the training features. *Note: Do not use MLP in sklearn for this HW - use Torch.*

a. Train your network for 100 epochs with a learning rate of 0.1 and batch size of 256 with the SGD optimizer. Record the training and validation loss after each epoch and compute the validation error of the final model. Report losses in the table at epochs 25, 50, 100.
b. Next, try different learning rates. Try at least three learning rates, e.g. 1, 0.01, and 0.001, and plot the training and validation losses. The `display_error_curves` function can be used to plot the losses.
c. Finally, see if you can improve the model by adjusting the learning rate, the hidden layer size, adding a hidden layer, or trying a different optimizer such as Adam (recommended). Report the train/val/test loss and the train/val/test classification error for the best model. You should be able to get a validation error less than 2.5%. Report your hyperparameters (network layers/size, optimizer type, learning rate, data augmentation, etc.).

3. **Species Prediction** [30 pts]

For this task, we use the Kaggle dataset Palmer Archipelago (Antarctica) penguin data. Your aim is to predict the species of penguin using features such as Culmen Length, Culmen depth, Body Mass, and Island.

We've cleaned the data for you (here) and turned strings into one-hot feature vectors, since `sklearn` doesn't handle categorical variables.

a. Feature Analysis

Spend some time to visualize different pairs of features and their relationships to the species. We've done one for you. Include in your report at least two other visualizations.

b. A Simple Rule

Suppose you want to be able to identify the Gentoo species with a simple rule with very high accuracy. Use a decision tree classifier to figure out such a rule that has only two checks (e.g. "mass greater than 4000 g, and culmen length less than 40 mm is Gentoo; otherwise, not"). You can use the library `DecisionTreeClassifier` with either 'gini' or 'entropy' criterion. Use `sklearn.tree.plot_tree` with feature_names and class_names arguments to visualize the decision tree. Include the tree that you used to find the rule in your report and the rule.

c. Design a ML model to maximize accuracy

Use any method at your disposal to achieve maximum 5-fold cross-validation accuracy on this problem. To keep it simple, we will use `sklearn.model_selection` to perform the cross-validation for us. Report your model design and 5-fold accuracy. It is possible to get more than 99% accuracy.

4. Stretch Goals

For (a), remember to design your model using only train and val, and evaluate once on the test set. These goals are meant to be open-ended challenges that require independent exploration, so not many hints will be given. Of these three problems, I would say (b) is the easiest, and (c) requires the most independent learning.

a. **Improve MNIST Classification Performance using MLPs** [up to 20 pts]

If you achieve low test error for part 2, you can claim additional points: +5 for test error < 2.3%, +5 for test error < 2.0%, +5 for test error < 1.8%, +5 for test error < 1.6%. You can also use an ensemble of networks to achieve lower error for this part. Describe your method and report your val/test error.

b. **Find a second rule** [5 pts]

Find a second rule to classify Gentoo that also requires only two checks but is different from the other one you reported.

c. **Positional encoding** [25 pts]

Use positional encoding to predict RGB values given pixel coordinate of this image. This has two parts. First, create a multilayer network that predicts RGB from (x,y). Then, create a multilayer network that predicts RGB from the sinusoidal positional encoding of (x,y). Show the images generated by each network. You can downsample the image if necessary (and probably a good idea to get it working with e.g. 32x32 image). See this paper for detail: https://arxiv.org/pdf/2006.10739.pdf. According to the paper, the design parameters are: L2 loss, ReLU MLP with 4 layers and 256 channels (nodes per layer), sigmoid activation on output, and 256 frequencies.

See this page for Python code of sinusoidal positional encoding: https://towardsdatascience.com/master-positional-encoding-part-i-63c05d90a0c3

Submission Instructions

Submit three files: (1) completed report template (pdf); (2) an html or pdf version of your Jupyter notebook; and (3) the pynb or zip of all files needed to run your code. Be sure to include your name, number of points claimed, and acknowledgments in the report. **The report is your primary deliverable**. The notebook and code are included for verification/clarification by the grader.