CS 441

HW 4: Trees and MLPs

Due Date: April 1, 2024 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 (especially important for this HW)
- Starter code

These assignments are based on the lectures: Decision Trees - Feb 22 (1, 3, 4b), Ensembles - Feb 27 (1, 3c), and MLPs - Mar 19 (2, 4a, 4c). Most students should start working on this after the midterm.

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`

For each method, train one model using the training set and measure RMSE on the training and validation sets. Plot the max_depth vs RMSE for all methods 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. It can be run with CPU, but will be a few times slower. See Tips

for detailed guidance on this problem. Note that this problem may require tens of minutes of computation.

First, use **PyTorch** to implement a Multilayer Perceptron network with one hidden layer (size 64) with ReLU activation. Set the network to minimize cross-entropy loss, which is the negative log probability of the training labels given the training features. This objective function takes unnormalized logits as inputs. *Do not use MLP in sklearn for this HW - use Torch.*

a. Using the train/val split provided in the starter code, train your network for 100 epochs with learning rates of 0.01, 0.1, and 1. Use a batch size of 256 and the SGD optimizer. After each epoch, record the mean training and validation loss and compute the validation error of the final model. The mean validation loss should be computed after the epoch is complete. The mean training loss can either be computed after the epoch is complete, or, for efficiency, computed using the losses accumulated during the training of the epoch. Plot the training and validation losses using the `display_error_curves` function.
b. Based on the loss curves, select the learning rate and number of epochs that minimizes the validation loss. Retrain that model (if it's not stored), and *report training loss, validation loss, training error, validation error, and test error.* You should be able to get test error lower than 2.5%.

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. *In your report, include the tree, the rule, and its precision and recall.*

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 30 pts]

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. Report your hyperparameters (network layers/size, optimizer type, learning rate, data augmentation, etc.). You can also use an ensemble of networks to achieve lower error for this part. Describe your method and report your val/test error. You must select a model using the validation set and then test your selected model with the test set. Points are awarded as follows: +10 for test error < 2.2%, +10 for test error < 2.0%, +10 for test error < 1.8%.

b. Find a second rule [10 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 [30 pts]

Advanced

Because linear functions are easier to represent in MLPs, it can help to represent features in a way that makes them more useful linearly. An example is the use of positional encoding to represent a pixel position, as described in https://arxiv.org/pdf/2006.10739.pdf.

For this problem, use positional encoding to predict RGB values given pixel coordinate of this image. You can resize the image to a smaller size for speed (e.g. 64 pixels on a side). In this problem, the network is acting as a kind of encoder — you train it on the same pixels that you will use for prediction.

1. Create an MLP that predicts the RGB values of a pixel from its position (x,y). Display the RGB image generated by the network when it receives each pixel position as an input.

2. Write code to extract a sinusoidal positional encoding of (x, y). See this page for details.
3. Create an MLP that predicts a pixel's RGB values from its positional encoding of (x, y). Display the RGB image generated by the network when it receives each pixel position as an input.

The paper uses these MLP design parameters: L2 loss, ReLU MLP with 4 layers and 256 channels (nodes per layer), sigmoid activation on output, and 256 frequencies.

Submission Instructions

Append two files: (1) completed report template; (2) a pdf version of your Jupyter notebook. Be sure to include your name and acknowledgments. **The report is your primary deliverable**. **Be sure to select all relevant pages in Gradescope!** The notebook pdf is included for verification/clarification by the grader. Submit the combined file to Gradescope.

To create PDF of notebook: Use "jupyter nbconvert" -- see starter code.

To combine PDFs: use https://combinepdf.com/ or another free merge tool.