Lab One

MSDS Summer 2021

Note: Submit code via GitHub Classroom using Markdown Cells to **clearly** indicate which code answers which question and to answer short answer questions.

1. Download the Forest Cover Type Dataset by UCI Machine Learning from Kaggle. Create a dataframe with only the forests with cover type 1 or 2.

2. Do the following to get ready for training:

(a.) Create a Dataset class that has two outputs: all the numeric variables besides CoverType and the CoverType as 0 or 1 (corresponding to 1 and 2).

(b.) Randomly split your data into a training and validation Dataset object.

(c.) Create a DataLoader with whatever batch size you desire.

3. Set up the following:

(a.) A 3-layer Feed-Forward Neural Network for this data. Think about the size of the input/output layer. Only linear layers and activation functions are allowed right now!

(b.) An optimizer and appropriate loss function for binary classification.

4. Write a function that iterates over a dataloader, doing the following:

(a.) prints the average loss (average over each datapoint!),

(b.) has an option to update the parameters after each batch OR not

5. Write a loop that trains your model for ten epochs, and at the end of each epoch it prints the average loss on the training set and on the validation set.

6. Do at least 2 of the following:

(a.) Try training with a very high learning rate and plot the loss over time. Now try training with a very low learning rate and plot the loss over time. What do you observe?

(b.) Try training with different batch sizes and observe the results.

(c.) Create a NN architecture that clearly overfits the training data, now create one that severely underfits the data. How many parameters does each have? Is there a sweet spot?

(d.) Compare the confusion matrix of your best 3-layer NN with another method (i.e. logistic regression, random forest, etc.) for binary classification of tabular data.

(e.) Briefly explain what the idea behind Adam optimization is. Contrast this with another method.

OR do the following

(e.) Fix a small NN architecture and optimizer. Do different random seeds or learning rate converge to a similar performance, or is performance tied to the starting point/learning rate here? If the models converge to similar performance, are the model parameters converging to the same values or different values with the same performance?