ID6001W: Applied Deep Learning
Programming Homework 3

To be submitted by February 19, 2023 11:59 pm

1 Multilayer Perceptrons in PyTorch and Sklearn

1. This entire assignment must be completed in a jupyter notebook. Specifically, use Google Colab to do all your coding and then download the resultant notebook (with the outputs still in the cells) as an ipynb file and submit the same. Using Google Colab is recommended as it will ensure some level of uniformity in hardware and therefore ease the evaluation process.

2. We will be using sklearn and PyTorch for our neural networks. Using Tensorflow for neural networks is not allowed for this assignment.

3. For uniformity, please use this piece of code as the first cell of your notebook and make sure you run this cell before any other cells are run (If this cell is not present in your submission, you will be awarded 0 marks):

```
import random
import torch
import os
import numpy as np
def seed_everything(seed=42):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.use_deterministic_algorithms(True)
seed_everything()
```

4. In this assignment our goal is to code a multi layer perceptron including backpropagation and gradient descent using Pytorch. Since Pytorch takes care of backpropagation and gradient descent, we are primarily concerned with the code for forward pass and model building. We will not be using any Pytorch DataLoaders in this exercise.

5. The dataset to be used is the MNIST dataset. The dataset can be loaded with the following code:

```
import tensorflow as tf
(X_train, Y_train), (X_test, Y_test) = tf.keras.datasets.mnist.load_data()
```

6. Please split the X_train and Y_train data further using sklearn's train_test_split function with random_state as 42 and test_size as 0.20 to get the new X_train and Y_train and also X_val and Y_val.

7. Using matplotlib.pyplot.imshow, display the first image in the X_train numpy array and the corresponding label from the Y_train numpy array (This label should be printed in the next cell). Set the cmap parameter of this imshow function as "gray".

8. The network architecture is as follows:
   a. A Linear Layer with number of input features as 784 and number of output features as 16
   b. A Non Linear Activation of ReLU
   c. A Linear Layer with number of input features as 16 and number of output features as 32
   d. A Non Linear Activation of ReLU
   e. A Linear Layer with number of input features as 32 and number of output features as 16
   f. A Non Linear Activation of LeakyReLU
   g. A Linear Layer with number of input features as 16 and number of output features as 10
   h. An appropriate layer that finally gives us probabilities of the different classes. Choose this layer yourself. There is only one correct answer.

9. Make sure that you make a python class for the PyTorch model.

10. The optimizer for this exercise will be the Adam Optimizer. Keep the learning rate of the optimizer as 1e-2.

11. The loss function for this will be the CrossEntropyLoss already implemented in PyTorch. You should not write your own Cross Entropy Loss module.

12. Make a separate python class that will be used for training this model.Train this model for such that it sees each data point exactly 1000 times. Make sure you print the training loss every 50th time the full dataset is seen. If the the prints of your loss are not present, marks will be deducted.

13. Post training this model, use it to make predictions on all the three different sets of data we have which are the training set (X_train, Y_train), validation set (X_val, Y_val), testing set (X_test, Y_test).

14. Based on the predictions, find the precision, recall and f1 score for all the different classes (which are the digits). Also report the accuracies for each class and report a weighted average accuracy across all classes. Using any sklearn functions for this task is not allowed.

15. Now, we will do the same exercise with sklearn. Use the MLPClassifier class from sklearn. Using this class, keep the architecture of the hidden layers as the same as previously mentioned in point 8. Sklearn automatically adjusts and recognises the input sizes for the input and output layer of the neural network. Use relu as the activation for all layers and use the random_state value as 42. Do not change any other parameters of the class.

16. Fit the class to the X_train and Y_train data.

17. Use the fitted class to make predictions on the three datasets mentioned in point 13.

18. Create classification reports using sklearn.metrics.classification_report for the different sets of predictions.