DS 203 - Fall 2022 Deep Learning Assignment

(TAs – Ninad Gandhi (22m2151), Rishab Khantwal (180100095))

Go through these instructions carefully to avoid any confusions.
Deadline: November 6, 2022 11:59 PM

1. **Guidelines**

- This assignment must be done individually.
- Use python libraries such as numpy, matplotlib, and pandas.
- Use the markdown section in your notebook to specify which question you are attempting, so that it aids the TAs in checking.
- Also add comments and descriptive answer in markdown and alongside the code to convey your understanding about the problem!
- Cite sources and references wherever they are used and add a reference section at the end.

2. **Tutorials**

- Here is a link for a Pytorch tutorial.
- Stanford Lecture on deep learning: https://youtu.be/6SlgtELqOWc?t=3077
- Pytorch Documentation
- Jargon you should be comfortable with: Tensors, Optimizer, Models, Autograd or Backward Pass, Activations.
- Familiarize yourself with Huggingface library for DeepLearning. It will be used for the second problem in the assignment.

3. **Linear Regression using Neural Networks**

In this part of the assignment we will check the power of standard neural network layers to implement a given regression function. We will experiment with two functions

$$y_1(x) = \log(x + 0.1)\,, where\ x \sim U[0, 10]$$

$$y_2(x) = x^3 + 2x^2 - x - 2, where\ x \sim U[-2.5, 1]$$

a) **Data Generation [3]**

Firstly, generate 10K points in the given range using the following code:

```
np.random.seed(0)
```

```
X1 = np.random.uniform(size=10000)

X2 = 3.5*np.random.uniform(size = 10000) – 2.5
```

generate corresponding functions y1 and y2 as given above.

b) **Exercises [47]**

1. Write a Dataset module for the above dataset (3 sets train, validation and test, 70:15:15 respectively) - **[7]**
   - *Inherit from torch.utils.data.Dataset*
   - *Define __init__, __getitem__, __len__*
2. Define the Dataloader with batch_size of 16 - **[5]**
   - *Go through all the arguments of dataloader like drop_last, shuffle, batch_size*
3. Define the Neural Network Model - **[10]**
   - *Inherit from : torch.nn.Module*
   - *1 hidden layer, your choice of activation function after first layer and single numeric output.*
   - *Variable hidden layer size as input to model (4 as default hidden layer size)*
4. Define your loss function as torch.nn.MSELoss(). - **[2]**
   Explore other possible loss functions like Mean Squared Error, Huber Loss, or any other loss function you like. Describe why you think some loss functions work better than others?
5. Optimizer: Use SGD optimizer with learning rate of 1e-3 - **[2]**
   See zero_grad(), step()
6. Write the main training loop and validation loops for n epochs - **[6]**
   - *See loss.backward(), model.forward()*
   - *Use model.train(), model.eval(), torch.no grad()*
   - *Validate for each epoch, run for 100 epochs*
7. Plot the following (Re-initialize your models as and when required so that it doesn't inherit the already learned weights from older iterations): - **[15]**
   - *Training and Validation loss vs epoch in a single plot*
   - *Best Validation loss vs Hidden Layer size (use hidden size to be (2, 4, 6, 8, 10))*
   - *Compare the validation error for different activation functions in the hidden layer. For example, use ReLU, tanh, Sigmoid.*

- *Best Validation loss vs learning rate used (use learning rates in (1e-5, 1e-4, 1e-3, 1e-2, 1e-1) for max number of 20 epochs.*
- *Plot test set predictions after training your models against the actual function, for three different choices of hyper parameters. For example, in one plot you can choose LR = 1e-3, hidden_layer_size = 4, activation = Sigmoid, and so on. Report MSE for the same. (Use 20 epochs for these plots)*

4. **ATIS dataset - [50]**

   Fine-tune DistilBert on Intent classification

   Download the dataset from https://www.kaggle.com/datasets/hassanamin/atis-airlinetravelinformationsystem

   i. Install transformers library of huggingface.

   ii. Do label encodings of test labels. - **[5]**

   iii. Split the dataset into train and validation in the ratio 80:20 - **[5]**

   iv. `From transformers import DistilBertTokenizerFast` tokenizer of pretrained model "`distilbert-base-uncased`". Tokenize the text in the dataset using this tokenizer - **[10]**

   v. Inherit from torch.utils.data.Dataset
   Define `__init__,__getitem__,__len__` - **[5]**

   vi. `from transformers import DistilBertForSequenceClassification, Trainer, TrainingArguments`

   vii. Define appropriate training arguments as follows

```
num_train_epochs = 3
per_device_train_batch_size=16
per_device_eval_batch_size=64
evaluation_strategy = "epoch"
```

   viii. Study other hyper parameters in HuggingFace for fine tuning your implemented network. For example: weight decay, warmup ratio, max sequence length and others. How are these parameters affecting on training? - **[5]**

ix. Load `DistilBertForSequenceClassification from pretrained "distilbert-base-uncased"`

x. Give appropriate arguments to the Trainer: model, training_args, train_dataset, test_dataset

xi. Train the model using the trainer - **[5]**

xii. Evaluate the trained model and report final accuracy of your model. - **[5]**

xiii. Report train loss vs epoch, validation loss vs epoch, accuracy vs epoch in tabular format - **[10]**

5. **Submission Details**

- Create 2 notebooks for the assignment, namely <RollNo>_nn_regression.ipynb and ATIS_intent_classification.ipynb.
- For submission upload these notebooks as .pdf and .ipynb with code and plots intermixed (make sure to generate all the plots and report all numbers instead of just the providing code for the same).
- All plots should be clearly visible in the pdf file with appropriate titles for plots, axes and legends.
- Your submission should be a zip or tar.gz folder containing the following 4 files with the given naming convention:
- <RollNo>_nn_regression.pdf, <RollNo>_nn_regression.pdf, <RollNo>_ATIS_intent_classification.ipynb, and <RollNo>_ATIS_intent_classification.ipynb