Assignment 2
CS7.601 (Monsoon 2023)
Deep Learning: Theory and Practices
Submission Deadline:
11:55 PM, October 7th 2023

Max. Marks : 20

Instructions

1. Please submit your code for all the questions in **Jupyter Notebooks**. Analysis should be included in the notebook itself using markdown cells. **Note that unlike assignment 1, analysis and reasoning of the results obtained will carry a significant weightage in the marks distribution**.
2. For Q3, submit the trained encoder and decoder models as well.
3. Only the following libraries are allowed:
    - Numpy
    - Pandas
    - Matplotlib
    - Pytorch (version $\geq$ 2.0)
    - Tensorflow (version $\geq$ 2.0) (Only allowed for assignment 2, assignments 3 and 4 will be pytorch only)
    - tqdm
    - scikit-learn
4. **Only submissions made on Moodle will be considered for evaluations.**

1 RNN For Auto Regressive Models [3 Marks]

In this exercise you are asked to generate an Auto Regressive model and then create an RNN that predicts it. Generate samples of an Auto Regressive model of the form

$$X(t) = a_1 X(t-1) + a_2 X(t-2) + a_3 X(t-3) + U(t)$$

where $U(t) \sim \text{Uniform}(0, 0.1)$, $a_1 = 0.6, a_2 = -0.5, a_3 = -0.2$. Generate 2000 training and 2000 test examples using this model. Now train an RNN that predicts the sequence. Apply the training algorithm on new samples and calculate the averaged cost square error cost function.

1. Investigate RNN with 1,2 and 3 hidden layers. Describe the architecture used for the network.
2. Plot the epoch-MSE curve during training.
3. Report MSE (mean square error), MAE (mean absolute error) and R^2 (R-square) on the test data.

Reference : https://en.wikipedia.org/wiki/Coefficient_of_determination

2 Learning Long Term Dependencies [7 Marks]

There are $p+1$ input symbols denoted $a_1, a_2, \ldots, a_{p-1}, a_p = x, a_{p+1} = y$. a_i is represented by $p+1$ dimensional vector whose i^{th} component is 1 and all other are 0. A net with $p+1$ input units and $p+1$ output units sequentially observes input symbol sequences, one at a time, trying to predict the next symbols. Error signals occur at every single time steps. To emphasize the long term lag problem, we use a training set consisting of only two sets of sequences: $\{(x, a_{i_1}, a_{i_2}, \ldots, a_{i_{p-1}}, x) \mid 1 \leq i_1 \leq i_2 \leq \ldots \leq i_{p-1} \leq p-1\}$ and $\{(y, a_{i_1}, a_{i_2}, \ldots, a_{i_{p-1}}, y) \mid 1 \leq i_1 \leq i_2 \leq \ldots \leq i_{p-1} \leq p-1\}$. In this experiment take $p = 100$. The only totally predictable targets, however, are x and y, which occur at sequence ends. Training sequences are chosen randomly from the two sets with probability 0.5. **Compare how RNN and LSTM perform for this prediction problem.** Report the following.

1. Describe the architecture used for LSTM and for RNN. Also mention the activation functions, optimizer and other parameters you choose. Experiment around with multiple architectures and report your observations.
2. Plot the number of input sequences passed through the network versus training error (for both LSTM and RNN).
3. Once the training stops, generate 3000 sequences for test set.
4. Report the average number of wrong predictions on the test set in 10 different trials (for both LSTM and RNN).

Example : Consider $p = 5$. Suppose the set of random integers (i_1, i_2, i_3, i_4) are $\{1, 1, 3, 4\}$ respectively. Then, the sequence $\{(x, i_1, i_2, i_3, i_4, x)\}$ will have matrix representation as

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3 Encoder-Decoder architecture [10 marks]

You have to train an encoder-decoder model to learn a transformation which takes a sentence and outputs a transformed sentence. Both the sentence and the transformed sentence are 8 characters long and only consist of lowercase English alphabets. You have been provided with "train_data.csv" which consists of 7000 sentence-transformed sentence examples and "eval_data.csv" which consists of 2000 sentence-transformed sentence examples. Make note of the following points:

1. You are not supposed to use the sentences provided in "eval_data.csv" for training the model. You can use it to judge overfitting or for early stopping.

2. You are free to use whatever architecture you wish, i.e., LSTM vs RNN vs GRU, num of hidden units, num of layers, etc (**Attention networks and Transformers are not allowed**).

   - The only limitation is that the output of the encoder (the context vector) should be the final hidden state of your RNN/LSTM/GRU.
   - If using multiple layers, the final hidden state of any of the layers or any combination of them is acceptable as the output, eg: You can return the last layer's final hidden state, the first layer's final hidden state, concatenation of the first, second, fifth layer's final hidden states, sum of the first and the last hidden states, and so on.
   - There are no limitations on how the decoder uses the context vector.

3. You are free to train your model however you wish (Early stopping, dropout, optimizers, regularization, number of epochs, etc).

4. You are free to explore any other approaches to improve your model's performance as well (better embeddings, data preprocessing etc).

5. **You may want to explore the "teacher forcing" technique for training RNNs and embedding layers in pytorch/tensorflow**.

6. You do not need to provide any kind of analysis for this problem, but you are encouraged to conduct analysis of your own to find ways to improve your model.

7. **If something is present in your code, you should know how it works. You should have some idea of what the particular function does and how it achieves it. Saying you just copied it from the internet will not suffice**. During evals if the TAs determine you have simply copied some part of the code from the internet without understanding it, you may be penalized for this question.

3.1 Marks distribution

5 marks will be awarded for the code and 5 marks will be awarded for the performance of the model.

- 2 marks will be based on the performance of your model on "eval_data".
- 3 marks will be based on the performance of your model on unseen data consisting of 1000 examples.

A prediction will be considered correct if 6 or more characters are identified correctly. A prediction will be considered partially correct if 4 or 5 characters are predicted correctly. You will be awarded 1 point for every correct prediction and 0.5 points for every partially correct prediction.
Your marks for "eval_data" will be calculated by the formula:

$$\text{eval\_marks} = \min(2, \frac{score}{1400} \times 2)$$

Your marks for the unseen data will be calculated by the formula:

$$\text{marks} = \min(3, \frac{score}{700} \times 3)$$

Marks will be rounded to the nearest .5 marks. Use the code provided in "checker.py" to obtain the performance statistics for your code.

3.2 Additional instructions

1. **Save a encoder and decoder model after training. You have to submit these saved models in your submission as well.**
2. Both the encoder and decoder should have a function named predict.
3. The encoder's predict function should take a string as an input and return a tensor as the output.
4. The decoder's predict function should take a tensor as the input and return a string as the output.

5. decoder.predict(encoder.predict(input_string)) should output the model's prediction for "input_string".

6. Your model should work with the provided checker code.

7. Make sure you are able to load the saved models.