BLG561E Deep Learning
2020/2021 Fall
Assignment 3

- Please write your own codes, copying code parts from books, websites or any other source including your friends is considered as plagiarism.
- Do not upload your codes to any public platform (e.g. Github) until the deadline of homework passes.
- Do not forget to comment your code.
- Submit your source codes on Ninova before the deadline, late submissions and submissions via e-mail will not be accepted.
- If you have any questions, do not hesitate to send an email to guresti15@itu.edu.tr

1 RNN-LSTM-GRU (40 points)

For this part of the homework go to the blg561 directory and install the package via:

pip install -e .

RNN [10 points]

Implement a simple RNNLayer in bl561/layer/recurrent_layers.py . Learnable parameters are W_x, W_h and b which are set during initialization. Dimensions of parameters are given in comments. RNN layer should compute:

$h^{(t)} = tanh(b + W_h h^{t-1} + W_x x^t)$

LSTM [15 points]

Implement a simple LSTMLayer in bl561/layer/recurrent_layers.py . Learnable parameters are W_x, W_h and b which are set during initialization. Dimensions of parameters are given in comments. LSTM layer should compute:

$a = b + W_h h^{t-1} + W_x x^t$
$a = [a_i, a_f, a_o, a_g]$
$input = \sigma(a_i)$, $forget = \sigma(a_f)$, $output = \sigma(a_o)$, $input\_gate = tanh(a_g)$

$c^{(t)} = forget \odot c^{(t-1)} + input \odot input\_gate$
$h^{(t)} = output \odot tanh(c^{(t)})$

Note: forward function is used in order to obtain only hidden states for the input batch and it is assumed input batch is from the start of the sequence; therefore, cell state should be initialized to 0 and it is not necessary to return the resulting cell states.

GRU [15 points]

Implement a simple GRULayer in bl561/layer/recurrent_layers.py . Learnable parameters are W_x, W_h, b, W_{xi}, W_{hi} and bi which are set during initialization. Dimensions of parameters are given in comments. GRU layer should compute:

$a = b + W_h h^{t-1} + W_x x^t$
$a = [a_z, a_r]$
$update = \sigma(a_z)$, $reset = \sigma(a_r)$
$h_{candidate} = tanh(bi + W_{hi}(reset \odot h^{t-1}) + W_{xi} x^t)$
$h^{(t)} = update \odot h^{(t-1)} + (1 - update) \odot h_{candidate}$

After the implementation, test your codes with test_rnn.py, test_lstm.py, test_gru.py in the tests folder.

2 Language Modeling with Recurrent Neural Networks in PyTorch (45 pts)

In this part you are going to do a language modelling task with recurrent neural networks in PyTorch. You are expected to use Google Colab for this task.
3 options from the Project Gutenberg is provided as data: Pride and Prejudice, A Tale of Two Cities and The Brothers Karamazov.
You will use cross-entropy loss for optimization and use perplexity as a metric. You should get a maximum validation perplexity of 120 for Pride and Prejudice and 125 for the other books. Experimenting with only one book is sufficient. This is not a sufficient value for good language modelling but it is sufficient for this assignment. For a detailed explanation of cross-entropy and perplexity of a language model, check: Perplexity in Language Models
Follow the instructions in LanguageModelling.ipynb.

3 Extra Questions (15 pts)

Answer the questions in LanguageModelling.ipynb.