Machine Learning and Deep Learning I Homework 4

Instructor: Joonseok Lee

Deadline: 2022/12/18 Sun, 23:59

- No unapproved extension of deadline is allowed. Late submission will result in 0 credit.
- Optimize your code as much as you can. We do not guarantee to run unreasonably inefficient codes for grading. Remember, vectorization is important for efficient computation!
- You will be given skeleton files for doing your assignment. Detailed instructions are given in the comments below each method to complete. Please read them carefully before jumping into implementation!
- Each assignment is built and tested under Google Colaboratory. If you work on a local machine, you need to handle version issue on your own.
- Explicitly mention your collaborators or reference (e.g., website) if any. If we detect a copied code without reference, it will be treated as a serious violation of student code of conduct.

1 Recurrent Neural Networks [40 pts]

We are going to implement the recurrent neural network model from scratch. You have to work on the skeleton code `hw4_rnn.py` provided on the ETL.

(a) Complete the `collate_batch` function. [6 pts]
For more details, please refer to the inline comments of the skeleton code, lab slides and class materials.

(b) Fill in the the shapes of the parameters in `__init__(self, vocab_size, input_size, hidden_size, num_class)`. [3 pts]

(c) Implement `forward(self, inputs, length)`. Do **NOT** use pre-defined PyTorch layers. [10 pts]
For more details, please refer to the inline comments of the skeleton code, lab slides and class materials.

(d) Implement `compute_loss(self, prediction, label)` to compute the cross-entropy loss and count the number of correct predictions. Do **NOT** use loss function APIs provided by `torch.nn` library (ex. `torch.nn.CrossEntropyLoss()`). [7 pts]

(e) There has been minor changes with the model forward operation and loss computation. Check what has been changed from the train and evaluate functions that we have previously used, and complete the `train` and `evaluate` function that works for the current model architecture. [4 pts]

(f) When training your customized RNN model, search for the best model hyperparameters and try different combinations of training options such as optimizer types, learning rates and learning rate schedulers. Then visualize the experiment logs with TensorBoard. Upon successful training, the test accuracy should be above 70%. [10 pts]
For more details, please refer to the inline comments of the skeleton code, lab slides and class materials.

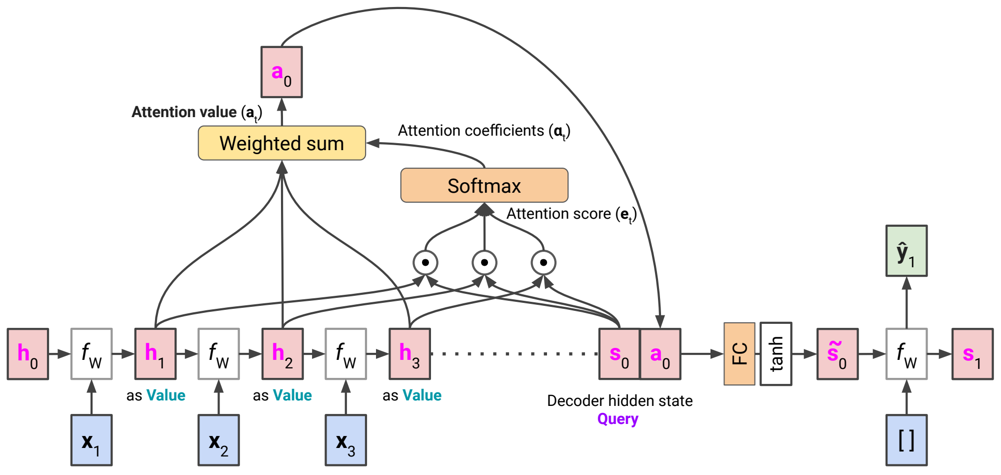

Figure 1: Attention Mechanism of encoder-decoder model, MLDL1 Lecture 9(Fall 22')

2 Translation with Attention Seq2Seq Model [30 (+10) pts]

In this exercise, we are going to implement an LSTM-based Seq2Seq translator with attention mechanism. In the decoder module, dot product attention should be implemented, according to Figure 1. If you want more detail, please refer to class and practice session materials, or original papers.

You have to work on the skeleton code `hw4_translation.py` provided on the ETL. Make sure to download the **data** folder, which includes skeleton code `data.py`.

(a) Implement `forward(self, ...)` with reference to the `__init__(self, ...)` for `LSTMEncoder` and `AttnLSTMDecoder` modules. For both Encoder and Decoder, what to be returned depends on your implementation of `LSTMSeq2Seq` (See Q2-(b)). So feel free to return any output you need. [10 pts]
For more details, please refer to the inline comments of the skeleton code, lab slides and class materials.

(b) Implement `forward(self, ...)` with reference to the `__init__(self, ...)` for `LSTMSeq2Seq`. Decoder module should attend encoder's outputs using dot product attention method. [10 pts]
For more details, please refer to the inline comments of the skeleton code, lab slides and class materials.

(c) Train your Seq2Seq model and plot perplexities and learning rates. Upon successful training, the test perplexity should be less than 7. Briefly report your hyperparameters and results on test dataset. Make sure your results are printed in your submitted file `hw4_translate.ipynb` [10 pts]

(d) Switch the modules(encoder, decoder) in Seq2Seq model from LSTM to GRU, and repeat Q2-(a), (b), and (c). Briefly report differeneces between LSTM and GRU based model.(*e.g.*, number of parameters, convergence rate, training results, etc.)[(Bonus) 10 pts]

3 Translation with Transformer Seq2Seq Model [30 pts]

In this exercise, we are going to implement a Transformer Seq2Seq model for translation tasks, based on the paper, Attention is All You Need (https://arxiv.org/abs/1706.03762). Dataset, translation task details, and skeleton code(`hw4_translation.py`) are the same as above(Q2). For more details on implementing transformer, please refer to the class material or original paper. (You are not allowed to use `torch.nn.Transformer` or other transformer libraries.)

(a) Implement `get_pos_emb(self)` in `TransSeq2Seq` module. Return an absolute positional embedding tensor based on `max_len` and `hid_dim` refering to equations below. [5 pts]

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

(b) Implement `make_src_mask(self, ...)` and `make_trg_mask(self, ...)` in `TransSeq2Seq` module. Return boolean padding mask tensors for both `make_src_mask` and `make_trg_mask`, and a boolean subsequent mask tensor for `make_src_mask`. [5 pts]
For more details, please refer to the inline comments of the skeleton code, lab slides and class materials.

(c) Implement `forward` methods in `TransEncoder`, `TransDecoder`, and `TransSeq2Seq` module, with reference to the `__init__(self, ...)` codes.
Use `torch.nn.TransformerEncoder/Layer` , `torch.nn.TransformerDecoder/Layer` to complete your code. Returns from `forwards` in `TransEncoder` and `TransDecoder` depend on your implementation of `TransSeq2Seq`. [10 pts]
For more details, please refer to the inline comments of the skeleton code, lab slides and class materials.

(d) Train your Seq2Seq model and plot perplexities and learning rates. Upon successful training, the test perplexity should be less than 2. Briefly report your hyperparmeters and results on test dataset. Make sure your results are printed in your submitted file `hw4_translate.ipynb`.
Also, based on the results from LSTM(GRU)-based and transformer-based Seq2Seq models, briefly describe which approach is better and why. [10 pts]

What to Submit

Please upload a single zip file named with your student ID (e.g., `2022-20000.zip`) on eTL, containing

- Your complete `hw4_rnn.ipynb` and `hw4_translate.ipynb` files.
- Please erase any unnecessary print codes or comments that you have wrote before submission.