STAT37710/CMSC35400 Homework 5

University of Chicago, Spring 2024 due Firday, May 17 at 11:59pm

In this assignment you will code a Long-Short Term Memory (LSTM) recurrent network in PyTorch to predict words in English language sentences taken from a story (some words, such as names, in the story have been modified). The network should have a single LSTM layer, and by default the h_t hidden vectors should be of dimension 200 (you may experiment with shorter or longer vectors). The c_t cell vectors must be of the same dimension.

You can set the input and output vectors to be also 200 dimensional, however size of the vocabulary, n_{vocab} is of course much greater than 200. To bridge this gap, try two different strategies:

1. Use a random word embedding, i.e., just assign a uniformly distributed random vector $v_i \in [0,1]^{200}$ to each word in the vocabulary.
2. Use a pre-computed embedding, as described on Martín Pellarolo's blog. You can find the embedding vectors here.

Please compare the accuracy of these two approaches.

Dataset

The data is provided in four files:

- `bobsue.seq2seq.train.tsv` is the training file consisting of 6036 pairs of consecutive sentences from the story separated by a tab.
- `bobsue.seq2seq.dev.tsv` is a validation set consisting of similar paired sentences that you can use to set parameters of the model and decide when to stop training (to avoid overfitting).
- `bobsue.seq2seq.test.tsv` is a test set of similar pairs of sentences.
- `bobsue.voc.txt` contains the vocabulary of the above files, with one word per line.

Each sentence in the training/validation/testing sets is enclosed between `<s>` `</s>` tokens.

Task

Your task is to learn to predict the second sentence of each pair, word by word, from the first sentence. In training, define the loss on each word as the squared distance $\| v - \widehat{v} \|^2$ between the predicted word vector and the vector corresponding to the correct word in the embedding space. At test time, just predict the word in the vocabulary whose vector is closest to the predicted word vector. You can ignore the `<s>` token, but please treat the `<s>` token as a word (or maybe an extra dimension of your output space), since predicting where the second sentence ends is highly significant.

Report the accuracy of your network on the test set as a function of various design parameters, and also plot how the loss on the training and validation sets changes as a function of training epochs. Report the 20 words that your network most often got right and the 20 words that it least often got right. Also include a sample of sentences produced by your network and the corresponding ground truth sentences. Submit the predictions of your network in a text file similar to the training/validation/testing files. Please do *not* submit the original training data.

In this assignment you are expected to use Pytorch's automatic differentiation functionality to compute all the gradients for you. You may use one of PyTorch's built-in training algorithms too (sugh as SGD etc) to update the weights. However, please do not use the pre-made LSTM class. An excellent tutrial on LSTMs can be found on Chris Olah's blog.