Deep Learning Assignment 4
LSTM

Meihao Chen
Center for Data Science
New York University
New York, NY 10003
mc5283@nyu.edu

1 Description of the Architecture

For character level RNN language model dataset I based the architecture on the model presented in *Recurrent Neural Network Regularization*, Wojciech Zaremba et al. 2014. The word level language model also follows the model in this paper.

1.1 Overall Structure

The architecture for the RNN language model is LSTM units with dropouts. I trained the character level RNN model in two sizes, as is in the Zaremba's paper. The hidden units are initialized to zero. I then use the final hidden states of the current minibatch as the initial hidden state of the following minibatch, whose size is 100.

1.2 Structure Details

The character level LSTM languge model has input sequence length of 50, and vocabulary size of 50. The LSTM has two layers and is unrolled for 35 steps. TThe best character model is the 'large size' model in Zaremba's paper, which has 1500 units per layer and its parameters are initialized uniformly in $[-0.04, 0.04]$. Non-recurrent connections have 0.65 dropout. The first 14 epochs have learning rate of 1, while later epochs reduce 1.15 after each. This model achieves 220.698 perplexity on validation set.

For the function query_sentences(), I trained a word level RNN language model, which has 200 nodes, 10000 vocabulary size, 50 sequence length, no dropouts, and achieved validation set perplexity of 105. I also train a baseline character level LSTM model using the same parameters except for the vocabulary size (50), and the validation perplexity is 282.

2 Description of the Learning Techniques Applied

The character level LSTM has 65% dropouts on RNN nodes, and the parameters are initialized uniformly in $[-0.04, 0.04]$. As is in Zaremba's paper, I clip the norm of the gradients at 10.

3 Description of the Training Procedure

The training parameters are as following:

- loss function: NLL
- optimization method: minibatch SGD
- learning rate: 1 and decay at 1.15 after 14 epochs

- batch size: 100
- learning rate decay: 1.15

4 Answers to the Questions

Q1 See $nngraph\_handin.lua$

Q2 $i = h_t^{l-1}$
$prev_c = c_{t-1}^l$
$prev_h = h_{t-1}^l$

Q3 create_network() returns a module that contains all layers in a timestep t. It is unrolled.

Q4 model.s stores all the memory on a certain timestep, and model.ds saves the gradient. model.start_s is the memory used on the start of each minibatch. If on the first minibatch of a timestop, model.start_s is set to zero, and the memory of the end of a minibatch is then set to be the model.start_s of next minibatch.

Q5 If the gradient norm is larger than maximum gradient norm, it is then shrinked by a factor to maximum norm.

Q6 Mini-batch SGD.

Q7 I set the extra output (the predicted values) to zeros when backpropogate, since they do not participate in gradient progcess.

5 References

Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. Recurrent Neural Network Regularization. *Journal of Machine Learning Research 15*,arXiv preprint arXiv:1409.2329 (2014).