Assignment 4: LSTM

Peter Varshavsky
pv629@nyu.edu

1 Answers to the questions

Q1. File `nngraph_handin.lua` provided in the git repo.

Q2. Relating lua variable names to equations in [1]

$$\texttt{i} = h_t^{l-1}$$
$$\texttt{prev\_c} = c_{t-1}$$
$$\texttt{prev\_h} = h_{t-1}^{l}$$

Q3. The function `create_network()` returns a GPU instance of one time slice of the LSTM network with number of layers specified by `params.layers`. Unrolling is done by `g_cloneManyTimes()` in `setup()`.

Q4. The table `model.s` keeps the states of the model at each layer and time step; `model.start_s` is used to reset the model's initial state to zeros at the beginning of forward propagation, and to reset it to the end-state of a sequence in case `seq_length` is exceeded; `model.ds` is used to allow the network to have a state output the same way as `dpred` is used in **Q7**.

Q5. Gradient is normalized as follows: if the matrix norm of the gradient matrix is greater than `params.max_norm` then the gradient matrix is multiplied by the shrink factor

$$\texttt{max\_norm}/\|gradient\|$$

Q6. Batch gradient descent is implemented in the back propagation through time function `bp()`. Runaway gradients are controled by putting a `max_grad_norm` limit on the gradients tensor, and learning rate annealing is implemented in the `main` loop after a specified number of epochs.

Q7. The extra output can be handled by passing a zero tensor of size `batch_size` × `vocab_size` to `backward()` function.

2 Network implementation

The submitted network is the small character-level baseline trained to 13 epochs. It consists of two hidden linear layers of size 200 and 20 time layers (`seq_length`) with LSTM cells and dropout implemented only on vertical (layer-to-layer, not time-instance-to-time-instance) connections as described in [1]. During training dropout was set to zero and not used. I apologize for a weak effort on this part of the assignment. The word-level and character-level networks predicted reasonable sequences using sequence generator functions in `main.lua`.

References

[1] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.