Deep Learning Assignment 4

Peter Yun-shao Sung yss265@nyu.edu

1 Warmup

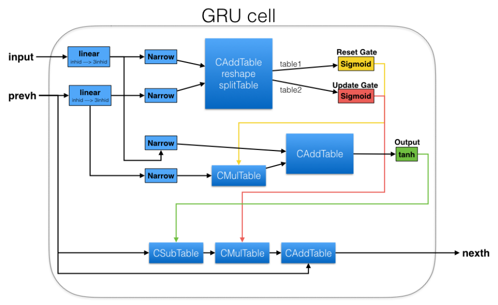

Figure 1: GRU cell unit

2 Approaches to RNN

2.1 Architecture

The initial model is based on lstm as building block, with 2 layers and 20 sequence length. Layers are horizontally considering the current given input (or word) and the states passed from previous sequence. Sequence is the stack of multiple layers, and is used for vertically passing states to the next layers. This way, model can not only predict the next word based on current word (horizontal), but the predictions can be altered based on the sequence of previous words (vertical). Regarding to each lstm layer, there are number of rnn_size lstm unit, and each of the unit perform the operation shown as figure 2.

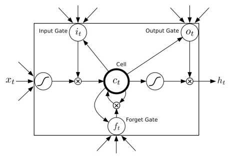

Figure 2: Operation of each of lstm unit

2.2 Learning Techniques

The two main learning techniques were the dropout and gradient clip. Dropout method was applied between each of lstm module and the final output. The inital module didnot used the dropout, and therefore we can observed that the perplexity of training improved well but not for the validation.

This implies the concerns of overfitting and dropout might be a good option to improve.
Gradient clipping might be important method during the learning process too. Soft-clipping was initially applied to the model, which it observes the L2 norm of gradient and every gradient will multiply the portions it overpassed if the norm is higher than the threshold. The other method I used during this assignment is hard clipping, which we only cut the gradiend for only the one that pass threshold, instead of the every gradients. The inspiration is mainly from the curiousity of whether soft clipping downgrade the learning process for other gradient. However, as designing gridient checker and tracking perplexity, I noticed simply cut the gradient that overpassed might be problematic, as there might be also many gradients need to cut and it turns out every gradients are at the boarder line, and this make generally high gradient and high perplexity.

2.3 Training Procedure

Data was preprocessed in the shape of (lines,batch_size), which is basically to traing a batch size of words. During each forward or backward propagation, line i and $i+1$ is the corresponding current and next word. When feed in network, the $next\_h$ is the output from the lstm module which is in the shape of (batch_size, vocab_size), and further calculates the prediction of each word along the batch_size axis by LogSoftMax. The perplexity is the accumulated error calculated by ClassNLLCriterion, and we are optimizing the perplexity error metric.

3 Improvement

The baseline perplexity is about 150, and here are the mothods I tryied to improve the prediction:

1. Clipping methods: The clipping method is hard clip as mentioned above, only cut the one that pass max gradient norm
2. GRUs: As it is a simpler method than lstm, here I include it for comparison as well
3. Hyperparameters: Including dropout rate, number of layers, soft/hard-clip, max gradient norm, number of sequence and rnn units

4 Results

The first test is about dropping rate. As we can noticed, if it's too high then we cannot lower the perplexity any further, but if it's too low, it is effective to improve the perplexity on training set but not for the test set, which is the concern for overfitting. Therefore, seems the dropout rate at the range between 40% to 50% can improve the overfitting issue as well as generate comparable result.

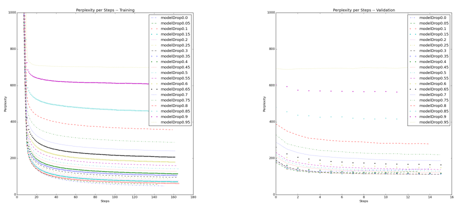

Figure 3: Parameter for dropout rate. Left is for training set. Right is for validation set

Second test is about the max gradient norm and its effect on soft or hard clip. There is a trend that the higher the value the later the perplexity goes down. Also the value I tested is higher than default(5), and seems higher value will cause more overfitting. Moreover, I tested the clipping methods on the dropout between 40% to 50%, which is the ideal range as mentioned ealier, seems hard clip have no way to further improve the perplexity any further.

The third test is implementing GRU as the unit cell. It is a simpler unit than lstm, and as using the default setting (20 seq_length, 200 rnn_size, 0 dropout, 2 layers), it can generate similar perplexity wich is around 200, but the overfittng seems more serious than lstm as unit cell. In lstm, the overfitting issue is the perplexity for training is keep lowering down while validation is just maintain in same level. However, in gru, perplexity for training is keep lowering down but validation is slightly going up which is bad. Probably having few more gates is necessary to maintain the state, and at least won't get worser perplexity when overfitting.

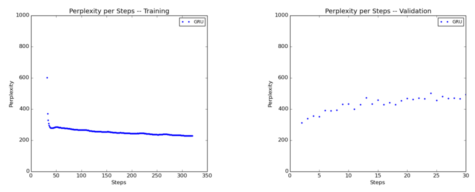

Figure 4: Model with GRU implemented. Left is for training set. Right is for validation set

Lastly, there are three parameters I think fall within the same idea that I would like to test, which are number of layers, number of rnn, and seq length. Each layer is the set of rnns, and seq length is the distant that I like to back propagate. They are all related to how am I going to hold the states and how to update it. Seems it gets harder to train when layer number increased (3 to 10 layers), and the perlexity is nothing better than baseline. The idea for more layer is to have more things to hold state, and therefore, in the other way around, I also did the test to decrease the number of layers but increase the rnn units. Surprisingly, the result is quite improved. The perplexity is improved from 120 baseline to 97 (0.4 dropout, 1 layer, 275 rnn, 20 seq length). Furthermore, as combining more rnn with higher seq length, the perplexity can goes down even further to 92, and, most importantly, the speed to reach this result can be around 13 epoch, while the previous is about 40 epoch. Althought it's not directly correspoing to actually time because it has higher rnn unit and seq to train, but the epoch difference is quite impressive.

Dropout	Layers	RNN units	seq_length	Train Perplexity	Test Perplexity
0.4	2	275	20	90.252	102.006
0.4	1	275	20	79.675	97.159
0.45	2	275	20	102.319	108.778
0.45	1	275	20	89.991	99.737
0.5	2	275	20	113.673	113.754
0.5	1	275	20	101.622	108.197
0.45	1	512	20	77.456	98.508
0.45	1	512	25	66.299	92.766
0.45	1	512	30	67.488	91.127

Table 1: Experiments on parameters and the perplexity

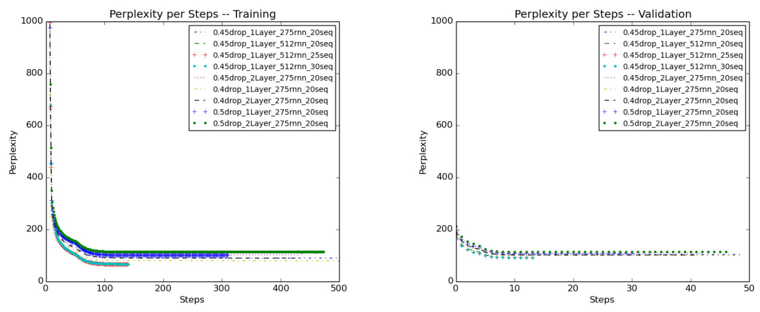

Figure 5: Parameter for dropout, layers, and rnn units. Left is for training set. Right is for validation set