Deep Learning Assignment IV: LSTM and RNN

Junbo Zhao (Jake)
CILVR Lab
CIMS, New York University
j.zhao@nyu.edu

Abstract

In this assignment, we investigate the Long Short Term Momery (LSTM) and Recurrant Neural Network (RNN) with Dropout. We follow the procedure of [1] and use their open-source code [2], delving into NLP problem.

1 Answer of question

- **Q1**
  See `nngraph_handin.lua` .
- **Q2**
  `i` corresponds to x_t input to LSTM cell at current time;
  `prev_c` corresponds to c_{t-1}, storing the previous LSTM cell state;
  `prev_h` corresponds to h_{t-1}, as the output of the LSTM cell at the previous stage.
- **Q3**
  `create_network` returns the rolled RNN, which will be cloned multiple times then for BPTT.
- **Q4**
  `model.s` stores all the LSTM cell states and outputs of each layer of RNN;
  `model.ds` is the gradient w.r.t the current cell state c_t and the consecutive output h_{t+1}, as the gradient yielded during one pass of propagation in BPTT;
  `model.start_s` stores LSTM cell states and outputs obtained from the previous sequence, also can be regarded as a pointer pointing to the current chunk of sequence. It is reset to 0 when one pass of training data is done.
- **Q5**
  By setting a maximum norm of gradient and multiplying a shrinking factor once the norm of the gradient is larger than that threshold. This would guarantee the gradients are constrained by a maximum norm.
- **Q6**
  Stochastic Gradient Descent is adopted, with learning rate decay.
- **Q7**
  We use `nn.Identity()(pred)` as an output node pulling out the predictions (in fact the log-probabilities). During the backprop pass, since this node shouldn't propagate any error, we set the corresponded `gradOutput` to 0.

2 Word-level model

We start with word-level models on PTB dataset. Due time constrain, we only experiment two models. The configurations and results are shown in Table. 1.

layers	rnn_size	dropout	vocab_size	max_grad_norm
2	200	0	10000	5
2	1500	0.65	10000	10

Train Perp	Valid Perp	Test Perp
39.522	120.395	115.870
35.594	82.280	79.488

Table.1 The performance of word-level model

The models are trained by SGD, with learning rate `lr=1`, without momentum. The error adopted is Negative Log-Likelihood (NLL). The train/valid/test dataset split is 42068/3370/3761. The perplexity is used as evaluation metric.

As we can see, the larger model with dropout performs better than smaller one. The `query_sentence` is implemented in `query_submit.lua`.

3 Character-level model

We then switch to Character-level model. As instructed, we change the `seq_length` to 50, in order to get longer sequence of characters being modeled. We also approximation perplexity by `exp(5.6 * mean(nll))` where we use 5.6 as the estimated length of word. Table. 2 reports the perplexities. Notive that all the Validation Perplexity reported in Table. 2 are obtained from calling `run_valid`. It slighlty differs from `a4_grading.py`.

layers	rnn_size	dropout	vocab_size	max_grad_norm
2	200	0	50	5
2	600	0.45	50	5
3	600	0.1	50	10
2	1500	0.5	50	10

Train Perplexity	Valid Perplexity
268.90	348.29
664.85	485.32
172.29	209.38
1063.15 (Unfinished)	579.36 (Unfinished)

Table.2 The performance of character-level model

Similarly, The models are trained by SGD, with learning rate `lr=1`, without momentum. The error adopted is Negative Log-Likelihood (NLL). The train/valid/test dataset split is 42068/3370/NA. The perplexity is used as evaluation metric. The motivation of the experiments shown in Table.2 are (1) Investigate the influence of model size; (2) Compare the models of different degree regularization (dropout).

As we can see, Table. 2 advices adding `rnn_size` and regularizing by dropout is able to improve the performance. However, adding dropout sometimes hurts the model performance clearly from the comparison of second and third row. Clearly, the training error of second experiment reveals an obvious underfitting since the training error becomes even larger than testing error. Meanwhile the best one (third) yields lower testing error than training error.

The best performance yielded so far is a slightly larger model (3-layer with `rnn_size=600`) with a reasonable dropout (`dropout=0.1`). From top three rows of Table.2, we reason that the model needs to be larger and deeper. In general, the testing error equals to training error plus generalization error; the dropout is proved to be able to control generalization error whereas testing error won't be decent unless if has low training error (ideally, 0). From the comparison we could see that training perplexity is still high. Therefore, we propose to use larger models. Because of time constrains, the larger one has not been finished. But we merely report the intermediate result of it.

In conclusion, we conjecture that only when the model gets into a regime of overfitting can dropout reacts by preventing the co-adaptation between units in the RNN.

References

1 Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. ”Recurrent neural network regularization.” arXiv preprint arXiv:1409.2329 (2014).

2 https://github.com/wojzaremba/lstm