DS-GA-1008: Deep Learning Spring 2015 Use Long Short-Term Memory for Word and Character Generation

Jiamin Xuan

April 28, 2015

Abstract

I use Long short-term memory(LSTM) Recurrent Neutral Network to generate next word based on some input words or characters on the Penn Treebank Wall Street Journal data with a larger model setting. The model achieves around 236 perplexity. Detailed setting and discussion are as belows.

1 Architecture

The character level model used a different setting from the baseline model. There are 2 layers in the model and the RNN size is increased to 1000. The batch size is 30 and sequence length is set to 35 and vocabulary length is 50. Regarding to learning techniques, a 0.65 dropout is used in the model and gradient clipping threshold is set to 9. Training uses stochastic gradient descent and learning rate starts at 0.5 and decays by 1.15. Parameters was uniformed with initial set of 0.04. Training/validation split at around 10%. There are 930,000 words in the training set and around 74,000 words in the validation set, in the data set, there are 10,000 different words has been used and 50 different characters. The data doesn‘t use much preprocessing techniques, just replaced space and taged the end of sentence. I think with more advanced preprocessing techniques we can achieve better score. Figure 1 shows the validation perplexity over 28 epoches, which achieve 236 at the 28 epoch.

2 Discussion

- Learning rate: conclude from word-level model when learning rate is set to 0.01, the model learns slow, in the baseline model, the learning rate was set to 1 which has extremely fast learning speed, training perplexity dropped hundreds of times at first. in the baseline model, learning rate decay is 2 which I changed to 1.15 in character level and set the initial rate to 0.5.
- Gradient clipping: gradient clipping is a simple way to control the optimization direction and its threshold could be set to the average of model gradient norm, in

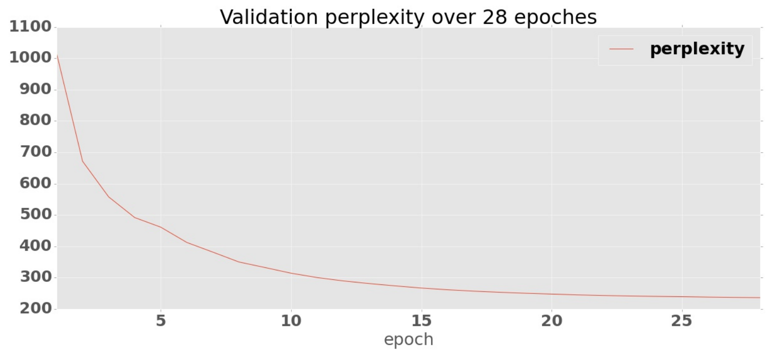

Figure 1: Validation perplexity over 28 epoches

the baseline model it is set to 5 which I modified to 9 according to the average gradient norm.

- Nesterov momentum: it is a simplified momentum update which is the same as the regular momentum, except for different coefficients. It is not implement in the current model but it could make the perplexity 10% lower at 40 epochs.
- Dropout: as described in Zarembas paper, dropout operator is only applied to the non-recurrent connections. As discovered in the word-level model, a 0.65 dropout could greatly reduce overfitting and contribute better prediction, in the word-level model, when dropout is turned off, the training perplexity reduced to 90 while validation perplexity goes back to 170 and after more epochs, the accuracy becomes even worse.
- Layer and LSTM unit: I didnt find a good way to tune these setting. When adding more layers the model performs worse than the 2 layers setting. I planned to perform a grid search of different setting but the training process for character level model cost around an hour for each epoch and theres some availability problem with GPU. Ill keep parameter tuning after submission and update the model if it is allowed.
- Word representation: input word is simply using a dictionary to index each word or character, and it just replaced space with _ and 'eos' tag for end of the sentence. Using GloVe or using convolutional layer to build word/character representation might be an interesting attempt. And more tags could be used to represent the structure.

3 Answers to the question

Q1: Please see nngraph_handin.lua in the git repo.
Q2: i=h_t^{l-1} , prev$_c$ =c$_{t-1}^l$, prev$_h$ =h$_{t-1}^l$

Q3: create_network() returns a rolled RNN and it is unrolled in setup() where it is copied many times.
Q4: model.s is the state of every step and model.ds is the gradient of the state, model.start_s is the initial state which reset to zero at the start of train, validation, testing, and when state.pos exceed the boundary of batches.
Q5: 2-norm
Q6: Stochastic gradient descent with annealing learning rate and gradient clipping.
Q7: Set gradient of prediction to zero so it won't affect the model in bp()

References

[1] Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850.*
[2] Zaremba, W., Sutskever, I., Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329.*
[3] Bengio, Y., Boulanger-Lewandowski, N., Pascanu, R. (2013, May). Advances in optimizing recurrent networks. In Acoustics, Speech and Signal Processing (ICASSP), *2013 IEEE International Conference on (pp. 8624-8628). IEEE.*