Note for *Mogrifier LSTM*

Ying Cao

February 28, 2020

Contents

Codes information

- Currently, the authors of this paper only release their experimental codes on the github.
- The final codes are not released yet. When the codes is available, it should be at https://github.com/deepmind/lamb.

1 Motivations

The author claims that domination of NLP by neural network models is hampered ***only*** by:

1. *their limited ability to generalize*
2. *questionable sample complexity*:
   (a) their poor grasp of grammar
   (b) their inability to chunk input sequences into meaningful units. While direct attacks on the latter are possible,

In this work, authors chose a natural language-agnostic approach to improve the generalization ablity of LSTM rather than directly attack the later since they believe the innovatioins in RNN architecture tend to have a trickle-down effect from language modeling to many other tasks.

While the LSTM is typically presented as a solution to the vanishing gradients problem, its gate i can also be interpreted as scaling the rows of weight matrices W^{j*} (ignoring the non-linearity in j).

2 Model

The standard LSTM update is a function:

$$\mathrm{LSTM}(\boldsymbol{x}, \boldsymbol{c}_{\text{prev}}, \boldsymbol{h}_{\text{prev}}) : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n$$

$$\mathrm{LSTM}(\boldsymbol{x}, \boldsymbol{c}_{\text{prev}}, \boldsymbol{h}_{\text{prev}}) = (\boldsymbol{c}, \boldsymbol{h})$$

.

Before the standard LSTM update taking place, a mogrifier is used. The mogrifier is essentially a gate prior to the input into each LSTM cell, and it is entirely based on the interaction between the hidden state and the input. In the mogrifier, $\boldsymbol{x}$ and $\boldsymbol{h}_{\text{prev}}$ modulate one another in an alternating fashion. See Figure 1 for this process.

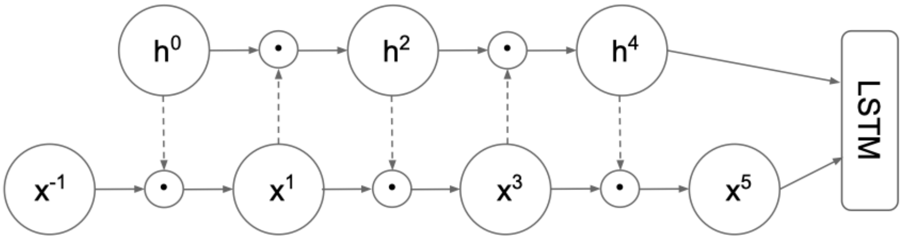

Figure 1: Mogrifier with 5 rounds of updates. The previous state $\boldsymbol{h}^0 = \boldsymbol{h}_{prev}$ is transformed linearly (dashed arrows), fed through a sigmoid and gates $\boldsymbol{x}^{-1} = \boldsymbol{x}$ in an elementwise manner producing $\boldsymbol{x}^1$. Conversely, the linearly transformed $\boldsymbol{x}^1$ gates $\boldsymbol{h}^0$ and produces $\boldsymbol{h}^2$. After a number of repetitions of this mutual gating cycle, the last values of $\boldsymbol{h}^*$ and $\boldsymbol{x}^*$ sequences are fed to an LSTM cell. The *prev* subscript of $\boldsymbol{h}$ is omitted to reduce clutter.

Figure 1: Mogrifier LSTM.

$\text{Mogrifier}(\boldsymbol{x}, \boldsymbol{c}_{\text{prev}}, \boldsymbol{h}_{\text{prev}}) = \text{LSTM}(\boldsymbol{x}^{\uparrow}, \boldsymbol{c}^{\uparrow}_{\text{prev}}, \boldsymbol{h}^{\uparrow}_{\text{prev}})$. $\boldsymbol{x}^{\uparrow}$ and $\boldsymbol{h}^{\uparrow}_{\text{prev}}$ are defined as the highest indexed x^i and $\boldsymbol{h}^i_{\text{prev}}$ as in equation(1) and (2), where $\boldsymbol{x}^{-1} = \boldsymbol{x}$ and $\boldsymbol{h}^0_{\text{prev}} = \boldsymbol{h}_{\text{prev}}$, $r \in \mathbb{R}$ is a hyperparameter. $r = 0$ recovers LSTM.

$$\boldsymbol{x}^i = 2\sigma(\boldsymbol{Q}^i \boldsymbol{h}^{i-1}_{\text{prev}}) \odot \boldsymbol{x}^{i-2}, \qquad \text{for odd } i \in [1 \dots r] \tag{1}$$

$$\boldsymbol{h}^i_{\text{prev}} = 2\sigma(\boldsymbol{R}^i \boldsymbol{x}^{i-1}) \odot \boldsymbol{h}^{i-2}_{\text{prev}}, \qquad \text{for even } i \in [1 \dots r] \tag{2}$$

3 Compare with other RNNs

Input Switched Affine Network [1]; Hypernetworks [2]; Multiplicative LSTM [3];

References

[1] J. N. Foerster, J. Gilmer, J. Sohl-Dickstein, J. Chorowski, and D. Sussillo, “Input switched affine networks: an rnn architecture designed for interpretability,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1136–1145, JMLR. org, 2017.

[2] D. Ha, A. Dai, and Q. V. Le, “Hypernetworks,” *arXiv preprint arXiv:1609.09106*, 2016.

[3] B. Krause, L. Lu, I. Murray, and S. Renals, “Multiplicative lstm for sequence modelling,” *arXiv preprint arXiv:1609.07959*, 2016.