The 32nd Annual Conference of the Japanese Society for Artificial Intelligence, 2018

Gated Recurrent Neural Network with Tensor Product

Andros Tjandra*1 Sakriani Sakti*1 Ruli Manurung*2 Mirna Adriani*2
Satoshi Nakamura*1

*1 Nara Institute of Science and Technology, Japan *2 Universitas Indonesia, Indonesia

In the machine learning fields, Recurrent Neural Network (RNNs) has become a primary choice for modeling sequential data such as text, speech, etc. To deal with long-term dependency in the long sequence, RNN utlizes gating mechanism to improve the gradient flow between multiple time-steps and avoid exploding/vanishing gradient problem. In the other hand, we would like to improve the representation power from RNN by using more expressive operation compared to standard matrix multiplication and summation. In this paper, we proposed a new RNN architecture with gating mechanism and tensor product between an input layer, a previous hidden layer, and a 3-rd rank tensor weight and we called it as gated recurrent neural tensor network (GRURNTN).

1. Introduction

Gating mechanism helps recurrent neural network (RNN) to model long-term dependency between each input time step. By using gating mechanism, RNN model able to avoid vanishing or exploding gradient [1]. On the other side, to help RNN model learn more complex dataset or task, we need to improve the expressiveness from our model. To get more powerful representation on hidden layer, Pascanu et al. [3] modified RNNs with an additional nonlinear layer from input to the hidden. Socher et al. [2] proposes another approach to model direct interaction between two input layers with tensor products on recursive neural network (RecNN). By using tensor product, we increase our RNN model expressiveness by using second-degree polynomial interactions, compared to first-degree polynomial interactions on standard dot product followed by addition in common RNNs architecture. In this paper we proposed a new RNN architecture that combine the gating mechanism and tensor product concepts to incorporate both advantages in a single architecture, denotes as gated recurrent neural tensor network (GRURNTN).

2. Proposed Approach

A gated recurrent unit (GRU) is a gated RNN with similar properties to a long short term memory (LSTM). However, there are several differences: a GRU does not have separated memory cells and instead of three gating layers, it only has two gating layers: reset gates and update gates. The GRU hidden layer at time t is defined by the following equations:

$$r_t = \sigma(x_t W_{xr} + h_{t-1} W_{hr} + b_r) \quad (1)$$

$$z_t = \sigma(x_t W_{xz} + h_{t-1} W_{hz} + b_r) \quad (2)$$

$$\tilde{h_t} = f(x_t W_{xh} + (r_t \odot h_{t-1}) W_{hh} + b_h) \quad (3)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t} \quad (4)$$

Contact: Andros Tjandra, Nara Institute of Science and Technology, 8916-5 Takayama-cho, Ikoma, Nara 630-0192, JAPAN, 0743-72-5265, andros.tjandra.ai6@is.naist.jp

where $\sigma(\cdot)$ is a sigmoid activation function, r_t, z_t are reset and update gates, $\tilde{h_t}$ is the candidate hidden layer values and h_t is the hidden layer values at time-t. In spite of having one fewer gating layer, the GRU can match LSTM's performance and its convergence speed convergence sometimes outperformed LSTM.

In our proposed architecture, the tensor product operation is applied between the current input and previous hidden layer multiplied by the reset gates for calculating the current candidate hidden layer values. The calculation is parameterized by tensor weight. To construct a GRURNTN, we defined the formulation as:

$$\tilde{h_t} = f\left(\begin{bmatrix} x_t & (r \odot h_{t-1}) \end{bmatrix} W_{tsr}^{[1:d]} \begin{bmatrix} x_t \\ (r \odot h_{t-1}) \end{bmatrix} + x_t W_{xh} + (r_t \odot h_{t-1}) W_{hh} + b_h\right) \quad (5)$$

where $W_{tsr}^{[1:d]} \in \mathbb{R}^{(i+d)\times(i+d)\times d}$ is a tensor weight for mapping the tensor product between the input-hidden layer, i is the input layer size, and d is the hidden layer size. Alternatively, in this paper we use a simpler bilinear form for calculating $\tilde{h_t}$:

$$\tilde{h_t} = f\left(\begin{bmatrix} x_t \end{bmatrix} W_{tsr}^{[1:d]} \begin{bmatrix} (r_t \odot h_{t-1}) \end{bmatrix}^\intercal + x_t W_{xh} + (r_t \odot h_{t-1}) W_{hh} + b_h\right) \quad (6)$$

where $W_{tsr}^{[i:d]} \in \mathbb{R}^{i\times d\times d}$ is a tensor weight. Each slice $W_{tsr}^{[i]}$ is a matrix $\mathbb{R}^{i\times d}$. The advantage of this asymmetric version is that we can still maintain the interaction between the input and hidden layers through a bilinear form. We reduce the number of parameters from the original neural tensor network formulation by using this asymmetric version. Fig. 1 visualizes the $\tilde{h_t}$ calculation in more detail.

3. Experiment

We used a PennTreeBank (PTB) corpus, which is a standard benchmark corpus for statistical language modeling. A PTB corpus is a subset of the Wall Street Journal (WSJ) corpus. In this experiment, we followed the standard preprocessing step that was done by previous research [4].

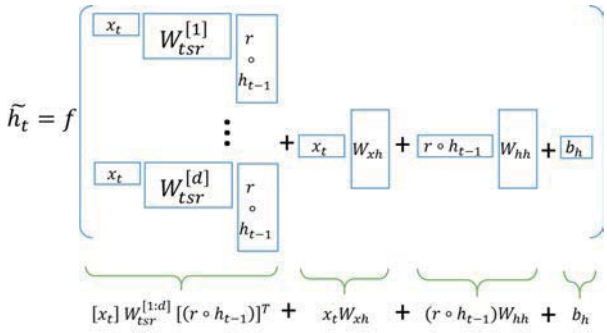

Figure 1: Calculating candidate hidden layer $\tilde{h_t}$ from current input x_t and previous hidden layer multiplied by reset gate $r \cdot h_{t_1}$ based on Eq. 6.

We used the preprocessed PTB corpus from the RNNLM-toolkit website [*1].

We did two different language modeling tasks. First, we experimented on a word-level language model and we used perplexity (PPL) to measure our RNN performance for word-level language modeling

$$PPL = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 P(X_i|X_{1..i-1})}. \quad (7)$$

Second, we experimented on a character-level language model and we used the average number of bits-per-character (BPC) to measure our RNN performance for character-level language modeling

$$BPC = -\frac{1}{N}\left(\sum_{i=1}^{N}\log_2 p(X_i|X_{1..i-1})\right). \quad (8)$$

3.1 Experiment Models

In this experiment, we compared the performance from our baseline models GRURNN with our proposed GRURNTN. For the word-level language modeling task, we used 256 hidden units for GRURNTN and 860 for GRURNN. All of these models use 128 dimensions for word embedding. The total number of free parameters for GRURNN and GRURNTN were about 12 million. For the character-level language modeling task, we used 256 hidden units for GRURNTN and 820 for GRURNN. All of these models used 32 dimensions for character embedding. The total number of free parameters for GRURNN and GRURNTN was about 2.2 million. We constrained our baseline GRURNN to have a similar number of parameters as the GRURNTN model for a fair comparison.

4. Result

In this section, we report our experiment results on PTB character-level and word-level language modeling using our baseline models GRURNN and our proposed GRURNTN. Table 1 shows BPC on PTB test set performed by our baseline model, our proposed model and several published results. Our proposed model GRURNTN and LSTMRNTN outperformed both baseline models. GRURNTN reduced the BPC from 1.39 to 1.33 (0.06 absolute / 4.32% relative BPC) from the baseline GRURNN. Overall, GRURNTN outperformed our baseline model and other published models on the character-level language modeling task.

Table 1: PennTreeBank test set BPC

Model	Test BPC
HF-MRNN [4]	1.41
sRNN [3]	1.41
DOT(S)-RNN [3]	1.39
GRURNN (our baseline)	1.39
GRURNTN (proposed)	1.33

Table 2: PennTreeBank test set PPL

Model	Test PPL
N-Gram [4]	141
RNNLM [4]	124.7
sRNN [3]	110.0
DOT(S)-RNN [3]	107.5
GRURNN (our baseline)	97.78
GRURNTN (proposed)	87.38

Table 2 shows the PPL on PTB test set performed by our baseline model, proposed model, and several published results. GRURNTN reduced the perplexity from 97.78 to 87.38 (10.4 absolute / 10.63% relative PPL) over the baseline GRURNN. Overall, GRURNN outperformed our baseline models as well as the other models significantly.

5. Conclusion

We presented a new RNN architecture by combining the gating mechanism and tensor product concepts. From our experiment on the PTB corpus, our proposed models outperformed the baseline models with a similar number of parameters in character-level language modeling and word-level language modeling tasks. In a character-level language modeling task, GRURNTN obtained 0.06 absolute BPC reduction over GRURNN (4.32% relative). In a word-level language modeling task, GRURNTN obtained 10.4 absolute PPL (10.63% relative) reduction over GRURNN.

6. Acknowledgment

Part of this work was supported by JSPS KAKENHI Grant Numbers JP17H06101 and JP17K00237

References

[1] Hochreiter, Sepp et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001).

[2] Socher, Richard et al. Reasoning with neural tensor networks for knowledge base completion. Advances in Neural Information Processing Systems (2013): 926-934

[3] Pascanu, Razvan et al. How to construct deep recurrent neural networks. arXiv preprint arXiv:1312.6026 (2013)

[4] Mikolov, Tom. Statistical language models based on neural networks (2012)

*1 http://www.rnnlm.org/