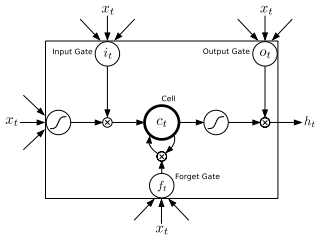
High Order LSTM/GRU

Wenjie Luo

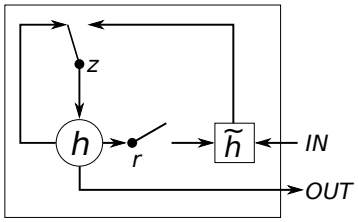
January 19, 2016

1 Introduction

RNN is a powerful model for sequence data but suffers from gradient vanishing and explosion, thus difficult to be trained to capture long range dependences. But people have proposed LSTM and GRU, which try to model the differences between adjacent data frame rather than the data frame itself. By doing so, it allows the error to back propagate throw longer time without vanishing. Also instead of simply accumulating information, these models introduce multiple gating functions which depend on current input and hidden states to control whether information should be taken-in, passed though or discarded. Currently all the gating functions are sigmoid/tanh function over a linear combination of input x and current hidden states h.
In this work, we are trying to introduce a high order term for RNN as a component of gating function. By doing so, we argue that it can better model the relation between current input and hidden states, thus better control the flow of information and learn better representation accordingly.



(a) LSTM[3]



(b) GRU[2]

2 Related Work

3 Model

3.1 LSTM

The basic building block of LSTM[3] is shown in Fig. 1a. It has three gates as shown below:

$$\begin{aligned} i_t &= \sigma(w_{xi}x_t + w_{hi}h_{t-1} + b_i) \\ f_t &= \sigma(w_{xf}x_t + w_{hf}h_{t-1} + b_f) \\ o_t &= \sigma(w_{xo}x_t + w_{ho}h_{t-1} + b_o) \end{aligned}$$

Correspondingly, the update equations are:

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \odot tanh(w_{xc}x_t + w_{hc}h_{t-1} + b_c) \\ h_t &= o_t \odot tanh(c_t) \end{aligned}$$

where $\odot$ is element-wise multiplication.

3.2 GRU

Alternatively, [1] developed another variant of RNN cell called gated recurrent unit. [2], [5] have shown this type of cell performs comparable/better than LSTM.
The basic building block of GRU is shown in Fig. 1b. Alternatively, the

update equations are as follows:

$$\begin{aligned} z_t &= \sigma(w_{xz}x_t + w_{hz}h_{t-1} + b_z) \\ r_t &= \sigma(w_{xr}x_t + w_{hr}h_{t-1} + b_r) \\ \widetilde{h}_t &= tanh(w_{xh}x_t + w_{hh}(r_t \odot h_{t-1}) + b_h) \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \widetilde{h}_t \end{aligned}$$

3.3 High order variant

In LSTM and GRU, the gate functions at time t are all sigmoid function over a linear combination of current input x_t and the memory represented via h_{t-1}. While gating functions are crucial for the network's performance as shown in [4], we further introduce a high order gating function:

$$g_t = \sigma(w_x x_t + w_h h_{t-1} + b_g + f(x_t, h_{t-1}))$$

where all vectors have dimension n. Here we only consider second order information. Assuming we are using m high order kernels, then

$$f(x_t, h_{t-1}) = W \begin{pmatrix} x_t^\top w_{xh}^{(1)} h_{t-1} \\ x_t^\top w_{xh}^{(2)} h_{t-1} \\ \vdots \\ x_t^\top w_{xh}^{(m)} h_{t-1} \end{pmatrix}, W \in \mathbb{R}^{n \times m} \tag{1}$$

where W is a mapping from m kernel output to a vector of dimension n as required.
If we use low rank approximation, i.e. $w_{xh}^{(i)} = \sum_{j=1}^{r} (v_j^{(i)})(u_j^{(i)})^\top$, we can rewrite each element in the high order term to be:

$$x_t^\top w_{xh}^{(i)} h_{t-1} = \sum_{j=1}^{r} (v_j^{(i)})^\top x_t \cdot (u_j^{(i)})^\top h_{t-1}$$

As we are learning distributed feature representation, it's reasonable to use $v_j^{(i)}$ same as $u_j^{(i)}$ in order to reduce the number of parameters, i.e. high order kernel weight matrices $w_{xh}^{(i)}$ are all systematic. Thus we have:

$$x_t^\top w_{xh}^{(i)} h_{t-1} = \langle V x_t, V h_{t-1} \rangle, V \in \mathbb{R}^{r \times n} \tag{2}$$

For each gating function, the number of parameters we introduced is $n * m + r * n * m$, in addition to linear part $2 * n * n + n$.

3.4 Alternative

Alternatively, as developed in [6], the high order term can be:

$$f(x_t, h_{t-1}) = W(Ux_t \odot Vh_{t-1}) \tag{3}$$

where $\odot$ represents for element-wise multiplication, and $U, V \in \mathbb{R}^{m \times n}$, $W \in \mathbb{R}^{n \times m}$. Correspondingly total number of parameters for each gating function is $n * m + 2 * n * m$ in addition to linear part $2 * n * n + n$.
The difference between Eq.3 and Eq.1, besides using different U and V, is that Eq.3 only uses *one* high kernel term whereas Eq.1 uses m high oder terms. But Eq.1 is NOT a general case for Eq.3. *Current experiment doesn't show much difference.*

3.5 MLP version

Also, we can have a multiple layer perceptron for modeling the transition between hidden states. Similar architectures have been developed in [7]

4 Learning

As shown in Eq.2, the high order term can be represented as a concatenation of a fully connected layer and a dot-product layer. Thus learning could also be done via standard back-propagation.

5 Experiments

RNN, with either LSTM/GRU function, has been apply to multiple sequence analysis tasks, for example machine translation[9], caption generation[10], video analysis[8].

5.1 Video Analysis

Our first experiment is on synthetic data, i.e. bouncing digits, as using in [8]. Each video sequence contains a number of frames showing the one or multiple digits bouncing inside a fixed window. We experiment on three different tasks. Given a few input video sequence, we are trying to construct the input sequences themselves, and also predicting some future sequences.

References

[1] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[3] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[4] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015.

[5] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350, 2015.

[6] Vincent Michalski, Roland Memisevic, and Kishore Konda. Modeling deep temporal dependencies with recurrent grammar cells"". In *Advances in neural information processing systems*, pages 1925–1933, 2014.

[7] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[8] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *arXiv preprint arXiv:1502.04681*, 2015.

[9] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[10] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*, 2014.