Using Fast Weights to Attend to the Recent Past

Zhang, Liqiang

May 23, 2018

Ordinary recurrent neural networks typically have two types of memory that have very different time scales, very different capacities and very different computational roles [3]. The history of the sequence currently being processed is stored in the hidden activity vector, which acts as a short-term memory that is updated at every time step. The capacity of this memory is $O(H)$ where H is the number of hidden units. Long-term memory about how to convert the current input and hidden vectors into the next hidden vector and a predicted output vector is stored in the weight matrices connecting the hidden units to themselves and to the inputs and outputs [1]. These matrices are typically updated at the end of a sequence and their capacity is $O(H2)+O(IH)+O(HO)$ where I and O are the numbers of input and output units [2].

The update rule for the fast memory weight matrix, A, is simply to multiply the current fast weights by a decay rate in Eq. 1, λ, and add the outer product of the hidden state vector, $h(t)$, multiplied by a learning rate,η:

$$A(t) = \lambda A(t-1) + \eta h(t)h(t)^T, \tag{1}$$

The next vector of hidden activities, $h(t+1)$, is computed in two steps. The "preliminary" vector $h_0(t+1)$ is determined by the combined effects of the input vector $x(t)$ and the previous hidden vector: $h_0(t+1) = f(Wh(t) + Cx(t))$, where W and C are slow weight matrices and $f(.)$ is the nonlinearity used by the hidden units in Eq. 2. The preliminary vector is then used to initiate an "inner loop" iterative process which runs for S steps and progressively changes the hidden state into $h(t+1) = h_S(t+1)$

$$h_{s+1}(t+1) = f([Wh(t) + Cx(t) + A(t)_s(t+1)), \tag{2}$$

where the terms in square brackets are the sustained boundary conditions. In a real neural net, A could be implemented by rapidly changing synapses but in a computer simulation that uses sequences which have fewer time steps than the dimensionality of h, A will be of less than full rank and it is more efficient to compute the term $A(t)hs(t+1)$ without ever computing the full fast weight matrix, A. Assuming A is 0 at the beginning of the sequence,

$$A(t) = \eta \sum_{\tau=1}^{\tau=t} \lambda^{t-\tau} h(\tau)h(\tau)^T, \tag{3}$$

$$A(t)h_s(t+1) + \eta \sum_{\tau=1}^{\tau=t} \lambda^{t-\tau} h(\tau)[h(\tau)^T h_s(t+1)], \tag{4}$$

In Eq. 3 and Eq. 4, the term in square brackets is just the scalar product of an earlier hidden state vector, $h(\tau)$, with the current hidden state vector, $hs(t+1)$, during the iterative inner loop. So at each iteration of the inner loop, the fast weight matrix is exactly equivalent to attending to past hidden vectors in proportion to their scalar product with the current hidden vector, weighted by a decay factor. During the inner loop iterations, attention will become more focussed on past hidden states that manage to attract the current hidden state.

Model	R=20	R=50	R=100
IRNN	62.11%	60.23%	0.34%
LSTM	60.81%	1.85%	0%
A-LSTM	60.13%	1.62%	0%
Fast weights	1.81%	0%	0%

Table 1: : Classification error rate comparison on the associative retrieval task.

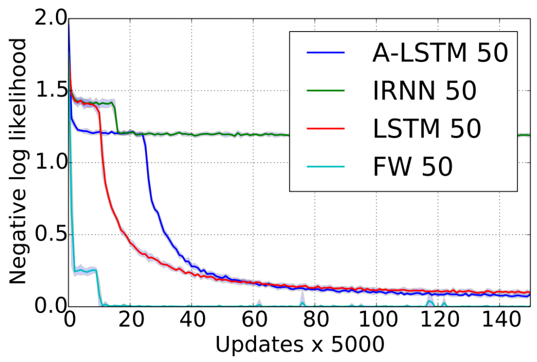

Figure 1: Comparison of the test log likelihood on the associative retrieval task with 50 recurrent hidden units.

Fig. 1 and Tabble. 1 show that when the number of recurrent units is small, the fast associative memory significantly outperforms the LSTMs with the same number of recurrent units. The result fits with our hypothesis that the fast associative memory allows the RNN to use its recurrent units more effectively. In addition to having higher retrieval accuracy, the model with fast weights also converges faster than the LSTM models.

References

[1] G Bi. Activity-induced synaptic modifications in hippocampal culture : dependence on spike timing, synaptic strength and cell type. *J Neuroscience*, 18, 1998.

[2] D. J. Willshaw, O. P. Buneman, and H. C. Longuethiggins. Non-holographic associative memory. *Nature*, 222(5197):960–962, 1969.

[3] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *Computer Science*, pages 2048–2057, 2015.