Fast Weights Using Improved Memory Consolidation Designs

Bowen Xu (bowenxu@cs.toronto.edu)
Department of Computer Science, University of Toronto
10 King's College Road, Rm. 3302, Toronto, Ontario, Canada, M5S 3G4

Jimmy Ba (jimmy@psi.toronto.edu)
Department of Electrical & Computer Engineering, University of Toronto
10 King's College Road, Rm. 3302, Toronto, Ontario, Canada, M5S 3G4

Richard Zemel (zemel@cs.toronto.edu)
Department of Computer Science, University of Toronto
10 King's College Road, Rm. 3302, Toronto, Ontario, Canada, M5S 3G4

Abstract

Storing better quality and greater amount of memory has been a difficult challenge for deep learning. While the current artificial neural networks excel at tasks such as regression and classification, they fail to perform equally well on representing variables and storing data over long time. In this paper, we introduce an enhanced fast weights model that includes robust memory consolidation designs into the existing memory transformation mechanisms. We show that our model comes with no additional costs, converges faster, and out-performs the original fast weights model on the associative retrieval tasks.

Keywords: Fast Weights; Memory Consolidation; Memory Transformation; Deep Learning; Associative Retrieval

Introduction

Retaining more memory for a longer period of time is an important goal for any memory system. Recently, Benna and Fusi have demonstrated that the combination of a power-law memory decay with a fast new memory adaptability maximizes both the memory storage and lifetime (2016). Their method treats each piece of memory as random and uncorrelated, and maximizes the signal to noise ratio (SNR) of all the stored memories (Benna & Fusi, 2016).

We adopt and adapt some of Benna and Fusi's ideas to current deep learning systems, notably the fast weights work (Ba, Hinton, Mnih, Leibo, & Ionescu, 2016). We show that our approach leads to a faster convergence and a stronger testing accuracy for the fast weights model on the associative retrieval tasks.

Background

Recent advances in deep learning (Graves et al., 2016; Ba, Hinton, et al., 2016) have shown that the use of an external memory with the artificial neural networks can significantly improve their abilities to retain information. Ba et. al. devise a bidirectional memory transformation system. Knowledge is not only transferred from the fast weights to the memory cells of a Recurrent Neural Network (RNN) through its hidden vector update but also transitioned from the memory cells to the fast weights as it is updated using the latest hidden vector at each time step. Also, a learning rate η and a decay rate λ are included to control how much the fast weights should store new knowledge versus how quickly it should forget previous knowledge.

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

$$h_s(t+1) = f(LN(Wh(t) + Cx(t) + A(t)h_{s-1}(t+1))) \tag{2}$$

$$h(t+1) = h_s(t+1) \tag{3}$$

$$w_a(t) \equiv \sum_{t'<t} \Delta w_a(t') r(t-t') \tag{4}$$

$$S_{t'}(t) \equiv \frac{1}{N} \left\langle \sum_{a=1}^{N} w_a(t) \Delta w_a(t') \right\rangle \tag{5}$$

$$N_{t'}^2(t) \equiv \left\langle \frac{1}{N^2} \left(\sum_{a=1}^{N} w_a(t) \Delta w_a(t') \right)^2 \right\rangle - S_{t'}^2(t) \tag{6}$$

$$S/N(t-t') = \sqrt{\frac{N r^2(t-t')}{\sum_{t"<t, t" \neq t'} r^2(t-t")}} \tag{7}$$

$$\sum_{t"<t, t" \neq t'} r^2(t-t") \approx \int_1^{\infty} r^2(t) dt \tag{8}$$

(1) shows that the fast weights $A(t)$ is updated using memory at the previous time step $A(t-1)$ and the outer-product of the current hidden vector $h(t)$. Knowledge stored in the fast weights is then transferred to the RNN's hidden vectors through *S* intermediate updates using (2). In (2), $h_s(t+1)$ is the current hidden state vector, $f(.)$ is the ReLU activation, LN is the layer normalization (Ba, Kiros, & Hinton, 2016), W is the weights for the RNN's hidden vector, C is the weights for the input to the RNN, $h(t)$ is the RNN's current hidden vector, $x(t)$ is the current input to the RNN, $A(t)$ is the current fast weights, and $h_{s-1}(t+1)$ is the hidden state vector at the previous intermediate update step. After *S* intermediate update steps, (3) shows that the last intermediate hidden state vector is assigned as the new hidden vector.

In Benna and Fusi's work, their memory is defined as the sum of the decayed synaptic modifications (4), their memory signal is defined as the averaged expected value of the sum of the overlap of each memory with its corresponding synaptic

modification given time t (5), their noise is defined as the square root of the signal variance (6), and their SNR is defined as the ratio of the memory signal over noise (7) (2016). From (7) and (8), we can see that in order to maximize the SNR, the integral in (8) must converge. Since Benna and Fusi use a power-law decay function which has the form of $t^{-\gamma}$, $\gamma > 0.5$ is required for the function to converge (2016). In addition, Benna and Fusi also suggest fast adaptability for the memory system which implies that the learning rate for new information should be exactly 1.0 (2016).

Model

We incorporate memory consolidation designs from (Benna & Fusi, 2016) to the fast weights model (Ba, Hinton, et al., 2016). We keep the fast weights setup and the weights transferring mechanism unchanged. However, our fast weights uses a learning rate of $\eta = 1.0$ instead of $\eta = 0.5$. We also substitute the original exponential decay function $\lambda = 0.95^t$ to a power-law decay function $\prod_{t=\tau}^{t=1} t^{-0.5}$ where τ is the number of time steps elapsed since the initial storage of a specific memory.

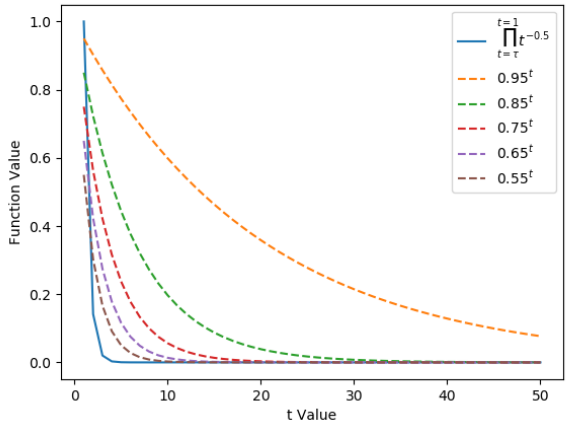

Figure 1: Decay Functions Comparisons

Figure 1 shows that our power-law decay function decays much faster than the original exponential decay function. We believe that because new information is stored in a greater portion than before, the fast weights has to decay its existing memory quicker to prevent memory interference. This design also forces the RNN to learn patterns quicker from the fast weights since existing information is quickly removed.

Experiments and Results

We evaluate three RNN models on the associative retrieval tasks on three difficulty levels. Our data consists of sequences of key-value pairs concatenated with two question marks and one query key. The keys are lower case characters from the English alphabet chosen randomly without replacement, and the values are digits among 0 to 9 chosen randomly with replacement.

Our training data contains $100,000$ sequences and our testing data contains $50,000$ sequences. The difficulty levels are three key-value pairs (K = 3), four key-value pairs (K = 4), and twenty-six key-value pairs (K = 26). The following models are evaluated on correctly predicting the associated value of the query key: a vanilla RNN (CONTROL), the original fast weights model (RNN-LN-FW), and the improved fast weights model (RNN-LN-FW2).

Table 1: Testing Accuracy of Different Models

Model	K = 3	K = 4	K = 26
RNN-LN-FW2	**99.74%**	**100%**	**100%**
RNN-LN-FW	99.5%	99.71%	94.75%
CONTROL	42.71%	34.49%	12.07%

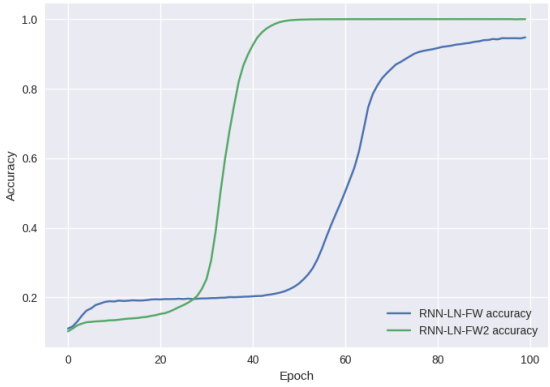

Figure 2: Testing Accuracy K = 26

Table 1 and Figure 2 show that our method leads to a faster convergence and a stronger testing accuracy for the fast Weights model.

Conclusion

We introduce an enhanced fast weights model by incorporating memory consolidation designs from (Benna & Fusi, 2016). We show that our model works better on the associative retrieval tasks and converges faster. Our next direction is to evaluate the models on more difficult tasks such as sorting and repeated-copying.

References

Ba, J., Hinton, G., Mnih, V., Leibo, J. Z., & Ionescu, C. (2016, October). Using Fast Weights to Attend to the Recent Past. *ArXiv e-prints*.

Ba, J., Kiros, J. R., & Hinton, G. E. (2016, July). Layer Normalization. *ArXiv e-prints*.

Benna, M., & Fusi, S. (2016). Computational principles of synaptic memory consolidation. *Nature Neuroscience*, *19*, 1697–1706.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwinska, A., ... Hassabis, D. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471–476.