CSE847 Project Intermediate Report

An analysis and verification of the efficacy of using Fast Weights with RNNs

Eric Alan Wayman
Department of Computer Science & Engineering
Michigan State University
waymane1@msu.edu

Adi Mathew
Department of Computer Science & Engineering
Michigan State University
mathewa6@msu.edu

ABSTRACT

This report describes our progress towards the goal of performing an analysis and reproduction of tests using the Fast Weights method (Ba et al. 2016a).

KEYWORDS

RNN, LSTM, Fast Weights, Memory

1 PROBLEM DESCRIPTION

Until recently, recurrent neural networks (RNNs), used for sequential processing, did not have a good way to share memory between sequences. The Fast Weights method introduced in the paper to be analyzed in this project (Ba et al. 2016a) addresses this limitation by providing the network with the capacity to store information about a given sequence during its duration to be used in the upcoming sequence. We will provide a full analysis and explanation of the methology, and replicate one of the empirical tests of the method, which compares its performance on an associative retrieval task to that of an iRNN and a long short-term memory network, or LSTM (Hochreiter and Schmidhuber 1997).

2 SURVEY OF PRIOR WORK

Recurrent neural networks (RNNs) are well-suited for learning from sequential data since weights are shared among different stages of the sequence (Goodfellow et al. 2016, p. 373). In particular, RNNs have been shown to perform well in tasks of speech-to-text conversion, creation of language models for both characters and words (Sutskever et al. 2011) and even frame by frame video analyses (Mnih et al. 2014). In RNNs, a given hidden state essentially acts as short-term memory for the previous state, determining the next hidden state together with the next input. One major issue in training RNNs with long input sequences is that the error gradients end up becoming very large or small (Schmidhuber 2015, p. 16) which implies that even if the network can be trained, the effect of an early hidden state on the current hidden state is practically non-existent. This problem was overcome by the introduction of the long short-term memory RNN (LSTM RNN), whose activation function has a constant derivative and thus does not explode or vanish (Schmidhuber 2015, p. 19). Unfortunately, the LSTM RNN's memory is still limited to an amount proportional to the number of hidden units in a sequence (Ba et al. 2016a, p. 1). Ba et al. propose the Fast Weights method to allow sequence-to-sequence memory in a recurrent network. We also note that Hopfield nets (MacKay 2002) implemented a similar storage rule (Ba et al. 2016a, p. 2) which we will review in our paper.

3 PRELIMINARY PLAN

Our term paper will first present the Fast Weights methodology and place it in the context of methods that led to its development. We will provide an extended description and derivation of the methodology for the purpose of verifying its properties. Our goal will be to also replicate Section 4.1 of the paper, which compares the Fast Weights' performance on an associative retrieval task with that of an Identity-RNN (iRNN) (Talathi and Vartak 2015), and an LSTM RNN (Ba et al. 2016a).

This project is intended to verify the foundational math and reasoning which justify the use of Fast Weights in a network. This will require us to retrace the work leading up to the introduction of Fast Weights. The initial stage of our project will be to perform a thorough proof and derivation of the equations for RNNs, and clearly explain the issues that led to the creation of LSTM RNNs. For instance, we will explain the "long-term memory issue" in RNNs beginning as follows. The expression of the hidden unit h_t at time t is:

$$h_t = g(\boldsymbol{W} \cdot x_t + \boldsymbol{U} \cdot h_{t-1} + b_h)$$

After t time steps, we get:

$$h_t = g(\boldsymbol{W} \cdot x_t + \boldsymbol{U} \cdot g(\cdots g(\boldsymbol{W} \cdot x_{t-T} + \boldsymbol{U} \cdot h_{t-T} + b_h) \cdots) + b_h)$$

Because of the T nested multiplications of h_{t-T} by $\boldsymbol{U}$, the effect of h_{t-T} on h_t is negligible (namely, the network does not have "long-term memory"). We will provide a full exposition of how this problem manifests itself when the network is trained.

The next stage of the project will involve explaining LSTM RNNs, their improvements over RNNs, and their limitations. We will then explain the mathematics of Fast Weights in RNNs, as well as several methodologies used in their implementation in the paper being studied such as layer normalization (Ba et al. 2016b), grid search (Goodfellow et al. 2016), and the Adam optimizer (Kingma and Ba 2014).

Following that, we will implement the Fast Associative Memory Network in MATLAB, and reproduce the analysis of Section 4.1 (Ba et al. 2016a) of the paper to confirm the performance of Fast Weights as compared to the iRNN and the LSTM RNN.

4 MEMORY AND RECURRENT NEURAL NETWORKS

Paragraph about the neural perspective.

Paragraph about treating as signals, etc.

There are two major ways of incorporating the temporal element of sequential data in neural network learning. One is providing a memory structure that can present data from multiple time

steps to a network that does not itself depend on the time variable (e.g. the time element is handled externally) (Haykin 2009, p. 672-673). The other is incorporating the time element directly inside the network via the use of feedback. Feedback occurs in a system when the output of an element of the system eventually affects the input to that same element (Haykin 2009, p. 18). There are two main types of feedback: local feedback and global feedback. Local feedback occurs when the output from an element directly feeds into that element's input, and global feedback occurs when the output eventually affects the input after passing through other elements first (Haykin 2009, p. 673).

Recurrent neural networks are networks containing at least one feedback loop of either type (Haykin 2009, p. 23). Feedback loops are inherently time-delay elements, where the output from an element is fed into a new element with a delay (in other words, transmission from one node to another is not instantaneous).

One major use of recurrent neural networks is to provide *associative memory*. In the storage phase, an associative memory is presented with key patterns and stores memorized patterns or values which are implicitly associated with their key patterns. In the recall phase, when presented with a distorted or incomplete version of a key pattern, the memory produces the associated value pattern (Haykin 2009, p. 38).

Let $s + l = N$, where s is the total number of neurons used for storage of patterns and l is the number of neurons that can be used for learning. One metric used when designing an associative memory is the ratio $q = \frac{s}{N} = \frac{s}{s+l}$. The total machine capacity (N) is limited to some value by resource constraints. Note that for a small q, the performance may be very good, though s is small relative to l: in other words, the network must use a large number of neurons l to recall very well only a few patterns. Therefore we seek to make q as large as possible while still achieving good performance, i.e. the network should be able to recall correctly many patterns by using as few neurons for learning as possible (Haykin 2009, p. 39).

REFERENCES

Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. 2016a. Using Fast Weights to Attend to the Recent Past. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 4331–4339. http://papers.nips.cc/paper/6057-using-fast-weights-to-attend-to-the-recent-past

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016b. Layer Normalization. *CoRR* abs/1607.06450 (2016). http://arxiv.org/abs/1607.06450

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Simon Haykin. 2009. *Neural Networks and Learning Machines (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. DOI:http://dx.doi.org/10.1162/neco.1997.9.8.1735

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). http://arxiv.org/abs/1412.6980

David J. C. MacKay. 2002. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and others. 2014. *Recurrent models of visual attention*. 2204–2212 pages.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117. DOI:http://dx.doi.org/10.1016/j.neunet.2014.09.003

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating Text with Recurrent Neural Networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. 1017–1024.

Sachin S. Talathi and Aniket Vartak. 2015. Improving performance of recurrent neural network with relu nonlinearity. *CoRR* abs/1511.03771 (2015). http://arxiv.org/abs/1511.03771

Table 1: *Project timeline*

Week	Dates	Task	Deliverable
Week 1	(2/6 - 2/13)	Background reading	
Week 2	(2/13 - 2/20)	Background reading	Proposal
Week 3	(2/20 - 2/27)	Data set construction	
Week 4	(2/27 - 3/6)	Background, foundational proofs	
Week 5	(3/6 - 3/13)	Partial implementation and preliminary run	
Week 6	(3/20 - 3/27)	Compose Intermediate Report	Intermediate Report
Week 7	(4/3 - 4/10)	Full implementation of networks	
Week 8	(4/10 - 4/17)	Complete empirical analysis	
Week 9	(4/17 - 4/24)	Compose Final Report	
Week 10	(4/24 - 5/1)	Finish Final Report and rehearse Presentation	Final Report Presentation