The Effects of the Memory Capacity of a Recurrent Neural Network for Time Series Prediction

Stow, May Tamara
Department of Computer Science
Federal University Otuoke, Bayelsa state, Nigeria

Article Information

Article # 01003
Received date: 4th Jan., 2020
Revision: 13th March, 2020.
Acceptance: 25th April, 2020
Published: 7th May, 2020

Key Words
Artificial neural network,
Memory capacity,
Recurrent neural network,
Time series prediction,
Real time recurrent learning

Abstract

The memory capacity of a recurrent network is the ability of a recurrent neural network to go back into the past and how far into the past the network can go to obtain input history to use to predict future output when performing a task. This study investigates the empirical and theoretical memory capacity of a fully connected and parametrized recurrent neural network. The rationale for doing this is to determine if the memory capacity of a recurrent neural network plays any significant role in the performance of the network. The data used in this study is the Santa Fe Laser dataset and the error function used is the Mean Square Error (MSE) function. The MSE at each epoch was calculated for the network before and after it was trained. A statistical analysis method known as the T-test was used to determine if there was a significant difference between the mean memory capacity of the network before and after it was trained. The main result of the study is that the theoretical memory capacity of a fully connected recurrent neural network does not in any way influence the performance of the network.

***Corresponding Author**: Stow, M.T.; maystow@gmail.com

Introduction

Artificial neural networks are a massively parallel distributed processor made up of simple processing units, which has a natural inclination for storing experiential knowledge and making it available for use. It resembles the brain in two aspects: Knowledge is acquired by the network from its environment through a learning process and inter-neuron connection strengths, known as synaptic weights, are used to store the acquired knowledge" (Haykin, 1999). Artificial Neural Networks are a major application of Artificial Intelligence. In the world today, artificial intelligence plays a significant role in many areas and works of life. The concept of artificial neural networks is being applied in many industries in the world today. The memory capacity and performance of these networks are important properties of the network and should be considered when using an artificial neural network to solve a real-world problem.

Several research papers have insinuated that the memory capacity of a recurrent neural network is an interesting property of the network. It is believed that having memory is a fundamental precondition for any recurrent neural network when performing an information processing task. (Jaeger, 2002) stated that the short-term memory capabilities of an echo state network enables it to model stochastic symbol processes generated by hidden-Markov models. This assumption implies that the memory capacity of a recurrent neural network enables it to perform a modeling task.

The investigation is concerned with the short-term and long-term memory effects in Recurrent Neural Networks. The short-term memory is memory effects connected with the transient activation dynamics of networks (Jaeger, 2002). The long-term memory is memory effects afforded by synaptic weight changes in learning, and memory effects brought about by switching phenomena in attractor dynamics (Jaeger, 2002).

Many tasks in Signal Analysis and Control require systems models with significant memory spans, i.e. where the system output should depend significantly on the input history (Jaeger, 2002). Neural networks were invented to solve artificial intelligence tasks without modeling a real biological system (Mezzano, 2007).

In modelling a system, the system should be able to use its memory to recall its previous outputs in order to determine its next output. This is why recurrent

neural networks were introduced. Various theoretical and empirical investigations on the memory capacity have been done. While evidence of the relationship between a recurrent neural network and the performance of the network has been established, no such relationship has been investigated theoretically between the memory capacity of a fully connected recurrent neural network, and the network before and after training in a task-solving setup Rodan and Tino (2011a), Rodan and Tino (2011b). Also, no investigation has been done on the relationship between a recurrent neural network before and after training, and their effect on its memory capacity.

Recurrent Neural Networks

Recurrent neural network is a kind of neural network with one or more feedback loops (Wang et al., 2002). The feedback loops can be from the output neurons to the input neurons, from the hidden neurons to the input neurons or from the hidden neurons back to themselves. The network's behavior is based on its history, and the presentation of patterns is represented in time (Mezzano, 2007). The feedback loops cause some delays in the network, and so they are referred to as delay units. The delay units provide the network with dynamic memory so as to encode the information contained in the sequence of units which is relevant to the expected output (Haykin, 1999). A major problem of recurrent networks is that training is very slow and requires a lot of computations. However, these networks have proven very useful in terms of predictive tasks.

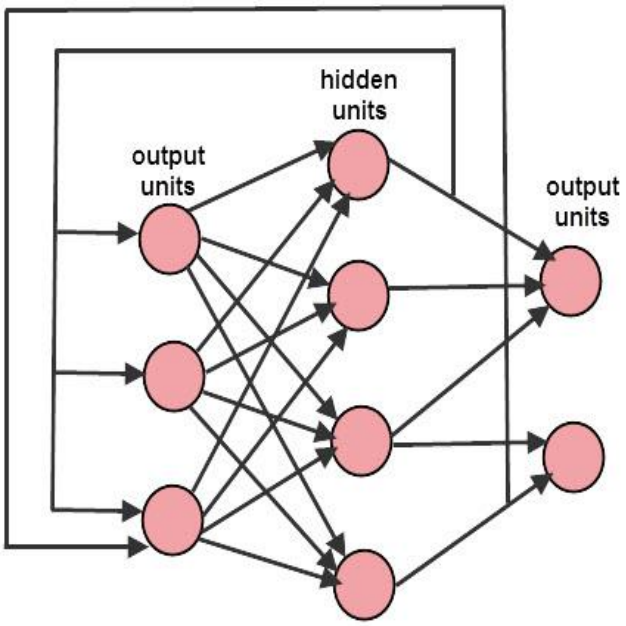

Figure 1: A simple recurrent neural network (Sinha *et al*., 2015).

Materials and Methods

Training of the Recurrent Neural Network

A fully connected recurrent neural network was used. The term "fully connected" means that all units (input, hidden and output units) are connected to each other. Training of the network was done with Real Time Recurrent Learning.

Real Time Recurrent Learning is a gradient-descent method which computes the exact error gradient at ever time step (Jaeger, 2010). Many authors, to their own understanding, have derived algorithms for local-time recurrent learning but the most cited reference for it is Williams and Zipser (1989). The algorithm derived its name from the fact that adjustments are made to the synaptic weights of a fully connected recurrent network in real time, that is while the network continues to perform its signal processing function (Haykin, 1999). Unlike Back Propagation Through Time (BPTT), no unfolding of the network is performed or necessary. In BPTT, there is this assumption that the weights remain fixed throughout the trajectory Williams and Zipser (1989). In RTRL, that assumption is relaxed and the weights are changed while the network is running has an important advantage that epoch (Epoch means a single pass through the entire training set, followed by testing of the validation set, ("Epoch in neural network," 2011) boundaries need to be defined for training the network, leading to both conceptual and

an implementational simplification of the procedure. The algorithm of RTRL is as follows:
The algorithm below, as stated in Williams and Zipser (1989) is for teacher-forced recurrent learning. Teacher-forcing is used in supervised learning tasks to replace the actual output $y_k(t)$ of a unit by the teacher signal $d_k(t)$ while modelling the network Williams and Zipser (1989).

$$z_k(t) = \begin{cases} x_k(t) & \text{if } k \in \text{I} \\ d_k(t) & \\ y_k(t) & \text{if } k \in \text{T(t)} \end{cases} \quad (1)$$

Where I is a set of inputs such that $\text{I} = \{x_k(t), 0 < k < m\}$, U is a set of hidden units or output units such that $\text{U} = \{y_k(t), 0 < k < n\}, n$ is the number of units in the network, m is the number of external input lines and $\text{T}(t)$ is a set of indices k in U for which $d_k(t)$ exists.

$$s_k(t) = \sum_{i \epsilon UUI} w_{kIZI}(t) \quad (2)$$

$$y_k(t+1) = f_k\big(s_k(t)\big) \quad (3)$$

The error of the output units is:

$$e_k(t) = \begin{cases} d_k(t) - y_k(t) & if\ k \in T(t) \\ 0 & otherwise \end{cases}$$

On that note, the error function for a single time step is:

$$E(r) = \frac{1}{2}\sum_{k \epsilon U} [e_k(\tau)]^2 \quad (4)$$

Assuming that the network is running from time t_0 to time t_1, the next step is to minimize the total error:

$$E_{total}(t_0, t_1) \sum_{t=t_0+1}^{t_1} E(\tau) \quad (5)$$

Where $\tau = t$.
Since the error is the sum of all the previous errors and the error at this time step, so also the gradient of the total error is the sum of the gradient for this step and the gradient for previous steps.

$$\delta_w E_{total}\big(t_0, t+1\big) = \delta_w E_{total}\big(t_0, t\big) + \delta w E(t+1) \quad (6)$$

As time series is fed to the network, the weight changes are accumulated. The weight change is calculated by the equation below:

$$\delta_{wij} = -u\frac{\delta E(t)}{\delta wij} \quad (7)$$

After all the time series has been fed to the network, each weight is now altered by the following equation:

$$\sum_{t=t_0+1}^{t_1} \partial_{wij}(t) \quad (8)$$

The next step is to derive an algorithm that computes the following equation:

$$-\frac{\partial E(t)}{\partial_{wij}} = -\sum_{k \epsilon U} \frac{\partial E(t)}{\partial y_k(t)} \frac{\partial y_k(t)}{\partial w_{ij}} \quad (9)$$

To derive a learning algorithm, we have to differentiate equation 9 with respect to $w_{ij.}$ Since $\frac{\vartheta d_i(t)}{\vartheta w_{ij}} = 0$ for all $leT(t)$ and for all t, the outcome is the equation below:

$$\frac{\partial y_k(t+1)}{\partial w_{ij}} = f'_k\big(s_k(t)\big)\left[\sum_{l \epsilon U - T(t)} w_{kl}\frac{\partial_{yl}(t)}{\partial_{wij}} + \delta_{ik} z_j(t)\right] \quad (10)$$

Where δ_{ik} is the Kronecker delta

$$\delta_{ik} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

Equation 10 is recursive, that is, if we know the value of the left-hand side for time 0, we can compute the value for time 1, and use that to compute the value for time 2, and so on ("Real time recurrent learning," 2011).

It is assumed that the initial state of the network has no functional dependence on the weights. Therefore, we have that

$$\frac{\partial_{yk}(t_0)}{\partial_{wij}} = 0 \qquad (12)$$

These equations hold for all $k\epsilon U, i\epsilon U, and\ j\epsilon U \cup I$.

The next step is to create a dynamical system with variables p_{ij}^k for all $k\epsilon U, i\epsilon U, and\ j\epsilon U \cup I$. The dynamical system is given as

$$p_{ij}^k(t+1) = f'_k\big(s_k(t)\big)\left[\sum_{l\epsilon U} w_{kl} p_{ij}^l(t) + \delta_{ik} z_j(t)\right] \qquad (13)$$

with initial conditions

$$p_{ij}^k(t_0) = 0 \qquad (14)$$

and it follows that

$$p_{ij}^k = \frac{\partial_{yk}(t)}{\partial_{wij}} \qquad (15)$$

For every time step t and appropriate i, j and k. The algorithm then consists of computing, at each time step t, the quantities p_{ij}^k and then using the error e_k to compute weight changes.

$$\delta_{wij}(t) = \mu \sum_{k\epsilon U} e_k(t)\, p_{ij}^k(t) \qquad (16)$$

The overall correction to be applied to each weight w_{ij} in the network is then simply the sum of these individual $\delta_{wij}(t)$ values for each time step t.

Theoretical Memory Capacity

The theoretical memory capacity introduced in Rodan and Tino (2011b), can be calculated using the following equations:

$$MC_k = V^T(W^k)G^{-1}W^kV \qquad (17)$$

G^{-1} is the inverse of the matrix G, V^T is the transpose of the matrix *V*, *V* is the input weights and *W* is the hidden or reservoir weights. The Matrix G can be considered as the covariance matrix of the iterated images of *V*, Rodan and Tino (2011b). The matrix G can be approximated by the following equation:

$$G = \sum_{l=0}^{\infty} W^l VV^T(W^T)^l \qquad fv \qquad (18)$$

The upper bound of l; L-max is approximated by the following equation:

$$\text{L\_}max = \frac{1}{2}\,\frac{\log\left(\frac{\epsilon(1-(\sigma_{max}(W))^2)}{N\|V\|_2^2}\right)}{\log(\sigma_{max}(W))} \qquad (19)$$

where ϵ is a precision term, $\sigma_{max}(W)$ is the largest singular value of *W* and $||V||_2^2$ is the squared Euclidean norm of *V*.

Results and Discussion

Fully connected generic recurrent neural network architecture of 2 input units, 50 hidden units and 200 output units was used for this experiment. The fully connected recurrent neural network was trained using real time recurrent learning as the learning algorithm, with Santa Fe Laser dataset. Sante Fe Laser dataset is a crosscut through periodic to chaotic intensity pulsations of a real laser Rodan and Tino (2011a), Schewenker and Labib (2009). The error function used was the Mean Square Error (MSE). While training, the task was to predict the next value *y (t + 1)*. The data contains 9000 values, the first 2000 values were used for training, the remaining 7000 were used for validation. The training was done for 10 epochs and the network after training was taken for the epoch with the least error. The test set MSE result of the network before and after training is shown in the table below.

The graph of the MSE for training at each epoch is shown below. During the network training, the networks before and after training were saved.

Table 1: Test set MSE results for RNN before and after training

Dataset	Network mode	MSE error
Laser dataset	Before training	0.203804
Laser Dataset	After training	0.000910

The hidden weights and input weights were extracted from the saved networks before and after training. After that, the hidden weights were used to replace the reservoir weights of the echo state network, and the memory capacity evaluated as usual. This was repeated 30 times (for 30 runs of input sequence) in other to have a stable memory capacity. The network before training had a better memory capacity than the network after training. This result is no surprise because that is always the case. Though the network before training had a better memory capacity than the network after training, this does not suggest that the network before training has a better predicting ability.

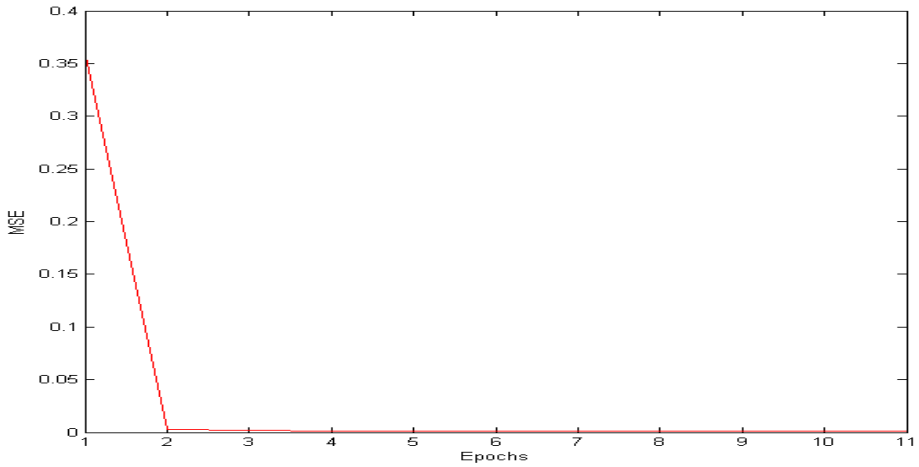

Figure 2: Graph of MSE for RNN training.

The 30 samples of memory capacity for the networks were used to perform a paired-sample t-test to see if there is any significant difference between the two samples. The results of the t-test are displayed in the table below.

Table 2: MC samples of RNN before and after training compared.

Recurrent neural network	p-value	h
RNN Before Training vs RNN After Training	2.2395e-34	1

From table 2, the p-value is less than 0.01 and the value of h is 1. This implies that there is a significant difference between the memory capacity samples of recurrent neural network before and after training. Since h is 1, it implies that null hypothesis is incorrect and there is a significant difference between the two samples (at the 5% significance level) that were compared.

Theoretical Memory Capacity Evaluation

The hidden weights and input weights which were extracted from the networks before and after training were used in calculating the theoretical memory capacity of a fully connected generic recurrent neural network. The hidden weights were scaled by

ISSN 2488-8648

multiplying the hidden weight matrix by the maximum singular value matrix. From the equation for MC theoretical, *V* is the input weights and *W* is the hidden weights. The number of delays *K* was chosen as 200 because values higher than 200 still gave the same result for *K* = 200. In other to appropriate *G*, the upper limit of *I* was estimated using the equation for *L_max* in equation 19. The value of e (precision for norms) was chosen to be 10e-60. The theoretical memory capacity was calculated using the hidden weights and input weights before training and was calculated using the hidden weights and input weights after training. The graph in figure 3 shows the results achieved.

Table 3: Theoretical MC of RNN before and after training

Recurrent neural network state	Memory capacity values
RNN Before Training	23.1526
RNN After Training	19.3954

From table 3, the theoretical memory capacity before training was better than the theoretical memory capacity after training. From figure 3, it is evident that the k-delay MC of the network before training starts falling earlier than the k-delay MC of the network after training. This implies that the network before training starts to forget its input history earlier than the network after training. The significance of this finding is that; although the theoretical memory capacity of the network before training is better than that of the network after training, its overall performance in time series prediction is not as good as that of the network after training. Therefore, it is safe to say that the performance of a network is not necessarily dependent on its memory capacity.

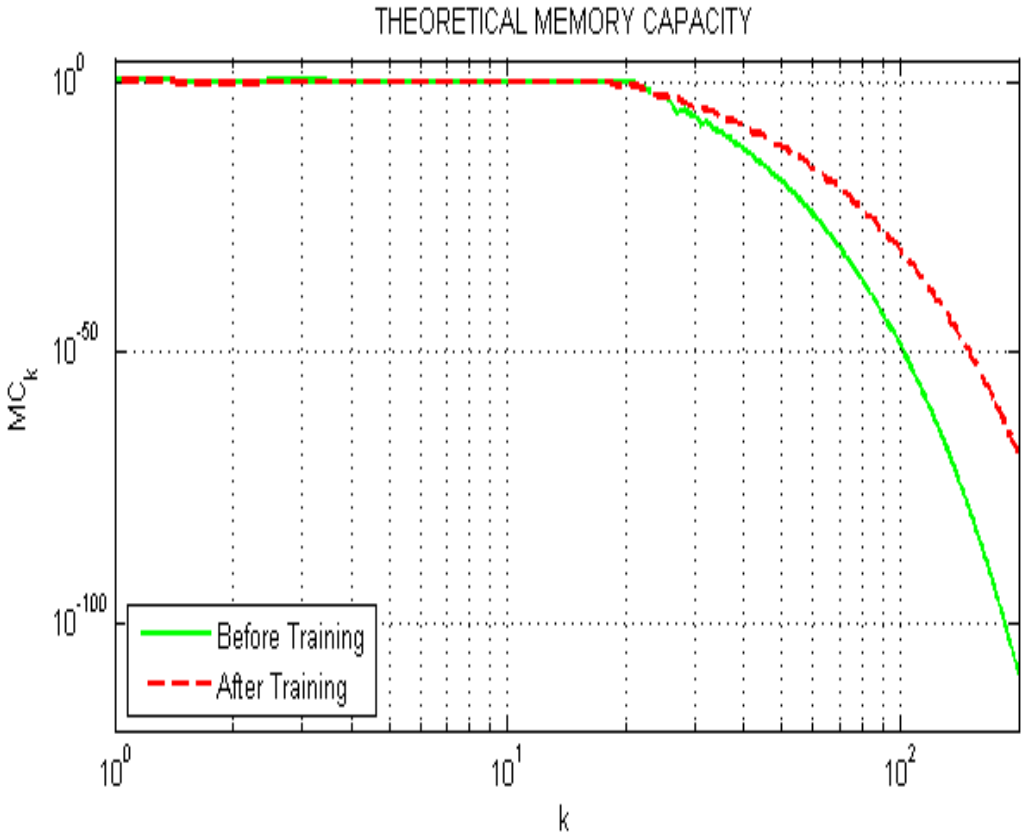

Figure 3: Theoretical Memory Capacity for a Recurrent Neural Network.

Conclusion

This study aims at investigating the memory capacity of Recurrent Neural Networks. The study has investigated the empirical and theoretical memory capacity of a fully connected recurrent neural network. This project was carried out to investigate whether or not the performance of a recurrent neural network is dependent on its memory capacity.

One of the more significant findings to emerge from this project is that theoretically, the memory capacity of a recurrent neural network does not play a role in the performance of the network. Neural networks are usually trained to learn a task; therefore, it is believed that a network performs better after it has been trained. The network before it was trained achieved a better memory capacity than the same network after it was trained. When the recurrent neural network was used for time series prediction, the results clearly showed that the network before training started forgetting its input history earlier than the network after training. Therefore, one of the evidences from this project suggests that the performance of a recurrent neural network does not depend on the network's memory capacity.

Directions for future work can include testing the recurrent neural network with other tasks such as speech recognition, pattern recognition, etc. Other learning algorithms can also be used to train the network to see if that has any impact on the way the network will perform when modelling a task.

ACKNOWLEDGMENT

The author is grateful to Peter Tino and Ali Rodan for their guidance during the course of this research.

REFERENCES

Epoch in neural networks. (2011). Retrieved from https://dictionary.babylon-software.com/epoch_in_neural_networks/

Haykin, S. (1999). Neural Networks: A Comprehensive Foundation, 1999 edn, Macmillian.

Hopfield, J. (1982), 'Neural networks and physical systems with emergent collective computational abilities', Proceedings of the National Academy of Sciences of the U.S.A 79, 2554–2558.

Jaeger, H. (2002), 'Short term memory in echo state networks'. volume 152 of GMD-Report, GMD-Forschungszentrum Informationstechnik.

Jaeger, H. (2010). 'The "echo" state approach to analyzing and training recurrent neural networks'. Technical report 148, German National Research Centre for Information Technology, St. Augustin, Germany.

Mezzano, T. (2007), 'Echo state networks: Application on maze problems'.

Miyoshi, S. and Nakayama, K. (1996). 'Probabilistic memory capacity of recurrent neural networks', Proceedings of the IEEE international Conference 2. Real time recurrent learning. (2011). Retrieved from https://www.willamette.edu/~gorr/classes/cs449/rtrl.html

Rodan, A. and Tino, P. (2011a). 'Cycle echo state reservoir with jumps.pp78.'

Rodan, A. and Tino, P. (2011b), 'Minimum complexity echo state network.pp65

Schweker, F. and Labib, A. (2009). 'Echo state networks and neural network ensembles to predict sunspots activity', Proceedings of the European Symposium on Artificial Neural Networks.

Sinha, R.P., Kesheri, M., Chowdhury, S. and Kanchan, S. (2015). Secondary and Tertiary structure Prediction of Proteins: A Bioinformatic approach. Complex System Modelling and Control Through Intelligent Soft Computations, pp. 541-569.

Wang, X., Lin, H., Lu, U. and Yahagi, T. (2002), 'Application of decision feedback recurrent neural network with real-time recurrent algorithm', Proceedings of the Power Conversion Conference on Neural networks 1.

Williams, R. and Zipser, D. (1989). 'A learning algorithm for continually running fully recurrent neural networks', Neural Computation 1(270-280).