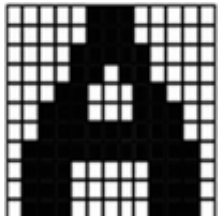
Crossroads
The ACM Student Magazine

# Storage Capacity Comparison of Neural Network Models for Memory Recall

by ***Kate Patterson***



## Abstract

The physiology of how the human brain recalls memories is not well understood. Neural networks have been used in an attempt to model this process.

Two types of networks have been used in several models of temporal sequence memory for simple sequences of randomly generated and also of structured patterns: auto- and hetero-associative networks. Previous work has shown that a model with coupled auto- and hetero-associative continuous attractor networks can robustly recall learned simple sequences. In this paper, we compare Hebbian learning and pseudo-inverse learning in a model for recalling temporal sequences in terms of their storage capacities. The pseudo-inverse learning method is shown to have a much higher storage capacity, making the new network model 700% more efficient by reducing calculations.

## Introduction

Memories stored in the brain can be recalled, among other forms, in temporal sequences. The memories tend to be associated in a way that recalling one memory triggers the recall of another memory. For example, one line or several lines from the lyrics of a song can trigger the recall of the next line. To more precisely describe the memory model examined here, consider a situation where a person is asked to remember a sequence of flash cards, then is shown one of the flash cards in a sequence and asked to recall the next card in the sequence.

Many computational models have been created to simulate temporal memory in the brain [2]. Neural network models, such as [6] and [7] by Lisman, have been proposed to simulate hippocampal activity, processes inside the temporal lobes of the brain. While the various neural network models may not be biologically realistic representations, they can be used to solve real-world problems such as predicting bankruptcy [8] or stock market fluctuations [4], with multilayer perceptrons. In general, optimizing the efficiency of network models makes the models more applicable for use with real-word applications.

In this paper, we examine recalled patterns or memory in a sequence. Specifically, the type of sequence considered is a simple sequence:

all patterns in the sequence are unique. Each pattern is denoted by $\xi$, where the $i^{th}$ pattern in a sequence is $\xi^i$. Given a pattern, $\xi^\mu$, the next pattern in the sequence, $\xi^{\mu+1}$, can be recalled if the sequence is simple. A simple sequence has no repeated patterns, so the next pattern in the sequence can be determined if the current pattern is known. Each pattern in a sequence has the same number of nodes, where $\xi_i^\mu$ is the $i^{th}$ node of the pattern $\mu$.

Two types of network connections are considered here: auto-associative and hetero-associative. An auto-associative network or module causes an input pattern to be pulled towards a recalled pattern that is similar to it. Once an input pattern in a hetero-associative network or module becomes similar enough to a pattern in the sequence, it is pushed towards the next pattern in the sequence. The input pattern is repeatedly passed between these two networks to progressively match the output pattern to each pattern in the sequence.

In order to make use of neural network models in real-world applications, the computational power and time required to run such simulations must be taken into consideration. The objective in this paper is to optimize previous work by Lawrence [**5**] in order to make its use more feasible. We consider specifically the storage capacity of a proposed network. The storage capacity, $\alpha$, is the maximum number of patterns, *p*, or the length of the sequence, that can be successfully recalled for a given number of nodes, *N*.

One factor affecting the number of patterns a network can store is the correlation of the patterns. Patterns that are randomly generated are typically uncorrelated [**9**]. If two or more patterns are quite similar, or correlated, then the model may relate the currently observed pattern with the incorrect next pattern. It becomes more likely that the auto-associative pull in the model will auto-associate to a close, but incorrect, pattern. A greater number of randomly generated patterns can be stored than structured or correlated patterns since randomly generated patterns are uncorrelated on average. The storage capacities of the models considered in this paper are calculated through testing on randomly generated patterns.

In a model proposed by Lawrence, Trappenberg, and Fine, two coupled, recurrent networks with auto- and hetero-associative connections are used to simulate the recall of simple temporal sequence memory [**5**]. The network contains two modules, the first of which outputs the next pattern in a sequence and the second of which outputs the previous pattern (Figure 1). A similar model was proposed by Jensen and Lisman [**3**]. In this model, an auto-associative network was coupled with a hetero-associative network, but it was limited to a very small number of patterns (approximately 4-7). Hence, the model is unable to handle more interesting pattern sequences or more realistic representations of a multitude of neurons contained in the brain. The ability to store many more patterns would give us more flexibility in model creation.

In the Lawrence-Trappenberg-Fine model, two auto-associative networks are coupled with hetero-associative connections. Module A and module B, as shown in Figure 1, are the auto-associative networks. The pattern stored in module B is used to hetero-associate the pattern in module A towards the next pattern. The remaining recurrent connections are used to auto-associate the current node values in each of the modules to the most similar pattern in the sequence.

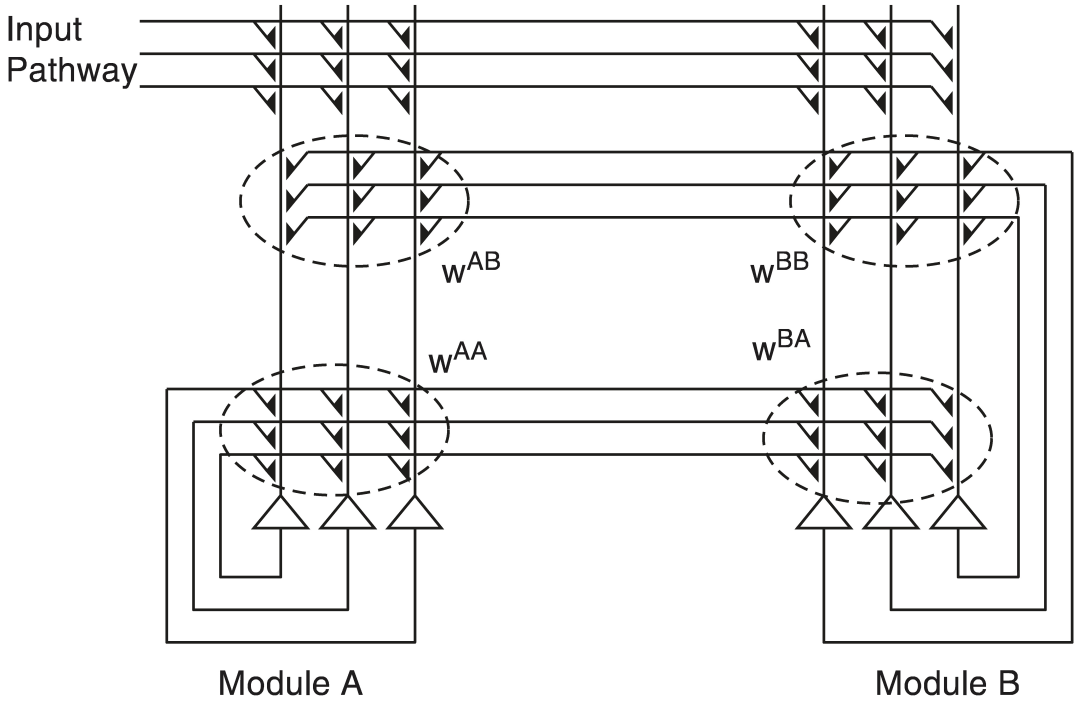



**Figure 1:** A pair of connected cortical modules, *A* and *B*.

In the figure, $w_{AA}$ and $w_{BB}$, and $w_{AB}$ and $w_{BA}$ represent the synaptic strengths in the recurrent connections in modules *A* and *B*, and the inter-module connections from *A* to *B* and from *B* to *A*, respectively [5]. $w_{AA}$, $w_{BB}$, and $w_{BA}$ are auto-associative weight matrices and $w_{AB}$ is a hetero-associative weight matrix.

## Recall

$$\tau \frac{dh_i^A(t)}{dt} = -h_i^A(t) + \lambda^{AA} \sum_{j=1}^{N} w_{ij}^{AA} r_j^A(t) + \lambda^{AB} \sum_{j=1}^{N} w_{ij}^{AB} r_j^B(t) \tag{1}$$

$$\mathrm{r}_i^A(t) = \tanh(h_i^A(t)) \tag{2}$$

$$\tau \frac{dh_i^B(t)}{dt} = -h_i^B(t) + \lambda^{BB} \sum_{j=1}^{N} w_{ij}^{BB} r_j^B(t) + \lambda^{BA} \sum_{j=1}^{N} w_{ij}^{BA} r_j^A(t) \tag{3}$$

$$r_i^B(t) = \tanh(h_i^B(t)) \tag{4}$$

Equations (1) through (4) are used to calculate and determine, over time, what the currently recalled pattern is.

The values of the nodes represent the input from the neurons in a brain. The weights of the connections represent the signal strength between neurons. As neurons fire, the strength with which they fire and the strength between connected neurons determines the resulting signals to the neurons. Equation (1) and Equation (3) are leaky integrator dynamics governing the activities of the nodes in modules A and B. $h_i^A$ and $h_i^B$ are the net inputs of the $i^{th}$ nodes in modules *A* and *B*. $\tau$ is the time scale. $\lambda^{AA}$, $\lambda^{AB}$, $\lambda^{BA}$, and $\lambda^{BB}$ are the path strengths from modules *A* to itself, *A* to *B*, *B* to *A*, and *B* to itself, respectively. $r_i^A$ and $r_i^B$ are the firing rates of the inputs to the $i^{th}$ nodes of modules *A* and *B*, respectively. The weights *w* are as previously described.

Initially, the weight matrices or synaptic strengths, $w_{AA}$, $w_{BB}$, $w_{AB}$ and $w_{BA}$, must be calculated using learning rules (see Learning Rules). Then the initial values of $h^A(0)$ and $h^B(0)$ must be set to the first pattern in the sequence. The network repeatedly calculates the firing rates (Equations (2) and (4)) and the change in net input (Equations (1) and (3)). $r^A$ is compared to the patterns in the sequence. When $r^A$ matches a pattern, this is considered a successful recall of that pattern.

## Learning Rules

### Hebbian Rule

$$w^A{}_{ij} = \frac{1}{N}\sum_{\mu} \xi_i^\mu \xi_j^\mu \tag{5}$$

$$P = \begin{bmatrix} \xi_1 & \xi_2 & \dots & \xi_p \end{bmatrix} \tag{6}$$

$$w^A = \frac{1}{N} P P^T \tag{7}$$

$$w^H{}_{ij} = \frac{1}{N}\sum_{\mu} \xi_i^{\mu+1} \xi_j^\mu \tag{8}$$

(9)

$$P_{rotate} = \begin{bmatrix} \xi_2 & \dots & \xi_p & \xi_1 \end{bmatrix}$$

$$w^H = \frac{1}{N} P_{rotate} P^T \tag{10}$$

The auto-associative weight matrix entries, $w_{ij}^{AA}$, $w_{ij}^{BA}$, and $w_{ij}^{BB}$, resulting from $p$ learned patterns are set equal to $w_i^A$ (5) [9]. This is the equivalent of creating a matrix consisting of the patterns as columns (Equation (6)), and calculating and setting $w^{AA}$, $w^{BA}$, and $w^{BB}$ equal to Equation (7).

Next, Equation (8) calculates hetero-associative weight matrix entries, $w_{ij}^{AB}$, resulting from $p$ learned patterns [9]. Similarly, create a matrix consisting of the patterns as columns rotated back one column as $P_{rotate}$ (9), calculating Equation (10) by multiplying the original pattern matrix with the transpose of the rotated pattern matrix [6]. Set $w^{AB}$ equal to this matrix.

Equation (5) calculates the $i^{th}$ column and the $j^{th}$ row of an auto-associative weight matrix. Equation (6) represents the matrix of patterns, where the $i^{th}$ column is the $j^{th}$ pattern in the sequence to be recalled. Equation (8) calculates the $i^{th}$ column and the $j^{th}$ row of a hetero-associative weight matrix. Equation (9) represents the matrix of patterns, where the $i^{th}$ column is the $i+1^{th}$ pattern in the sequence to be recalled. Equation (10) is a simplified equation for calculating the auto-associative weight matrix.

Note that while Equations (5), (7), (8), and (10) have a factor of $\frac{1}{N}$, the actual weight matrices in the program used to simulate the Lawrence-Trappenberg-Fine model do not have this factor, contrary to [5]. As such, this factor has not been used in either the Hebbian learning or the pseudo-inverse learning calculations.

The storage capacity of the Grossberg-Hopfield model [9], which uses the Hebbian learning rule, is $\alpha \approx .138$ [2].

**Pseudo-inverse Rule**

$$Q_{uv} = P^T P \tag{11}$$

$$Q_{uv}^{rotate} = P_{rotate}^T P \tag{12}$$

(13)

$$w_{inv}^{A} = P\left((Q_{uv})^{-1} P^{T}\right)$$

$$w_{inv}^{H} = P_{rotate}\left(\left(Q_{uv}^{rotate}\right)^{-1} P^{T}\right) \quad (14)$$

The auto-associative weight matrix, $w^{A}_{inv}$, resulting from *p* learned patterns is calculated using Equation (13). The pseudo-inverse learning rules have previously been used in auto-associated models and achieved a storage capacity of $\alpha \approx 1$ [1]. Given the improvement over the auto-associative storage capacity using Hebbian learning, we decided to apply this method to the Lawrence-Trappenberg-Fine model to see if the capacity could be improved.

The hetero-associative weight matrix, $w^{H}_{inv}$, resulting from *p* learned patterns is calculated using Equation (14). This method is problematic if $Q_{uv}$ is singular, because the inverse of a singular matrix cannot be calculated. Randomly generated patterns can create a singular *Q* and the probability is high for smaller values of *N*.

Applying the pseudo-inverse rule caused the resulting weight matrix entries to become much smaller. The change in net input to the modules, which depend on the weight matrices, decreases as a result. Consequently, $r^{A}$, which calculates *tanh* of the net input, would not reach the next pattern in the sequence. To compensate, the firing rates have been modified to be calculated as $r_i(t) = \tanh(\sigma * h_i(t))$. The value of $\sigma$ was tuned through trial and error to maximize the storage capacity.

## Experiment

### Comparing Randomly Generated Patterns to Bit-representations of Letters

In the first experiment, we test the recall accuracy of the Lawrence-Trappenberg-Fine model [1]. Using 156 nodes and 4 patterns, we used Lawrence's model on randomly generated patterns and then on representations of letters A, B, C, and D as demonstrated in Figure 2. The number of nodes was chosen to be 156 as the letters could conveniently be represented on a 12 by 13 grid of bits. The number of patterns was chosen to be 4 because of its low storage capacity requirement.

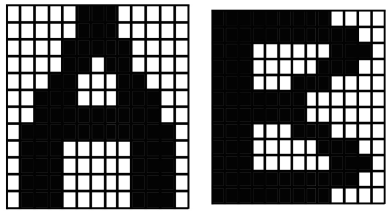


**Figure 2:** Bit representations of letters A and B.

The 12 by 13 nodes matrices are converted into 156 by 1 vectors (shown below A and B in Figure 2) to be used as input patterns. The conversion to a vector is performed by concatenating the first row with each of the consecutive rows.

### Comparing the Hebbian Learning Rule to the Pseudo-inverse Learning Rule

As previously mentioned, the storage capacity in an auto-associative network is greatly increased by using the pseudo-inverse learning rule instead of the Hebbian learning rule. We first ran an experiment using the Lawrence-Trappenberg-Fine model [**1**] on randomly generated patterns with varying numbers of nodes. We then ran the same experiment using the model with modified weight matrices that use the pseudo-inverse learning rule to determine whether or not pseudo-inverse learning can also improve the storage capacity of a network of coupled auto-associative modules with hetero-associative connections. Given that the network can fail to be created if the original $P^TP$ is singular, these occurrences were ignored during the experiment and tests were repeated until the number of required tests were completed. The experiments were run with $p = 2$ to $N$. For each pair of $N$ and $p$, the recall is calculated multiple times on different sets of randomly generated patterns. The accuracy is calculated as the number of successful recalls divided by the total number of tests on those values of $N$ and $p$. Conditions are detailed in Table 1.

**Table 1:** Conditions of the experiment.

| Experiment | Node range |
|---|---|
| Hebbian learning | $N$=2 to 350, $N$ increment by 15 |
| Pseudo-inverse | $N$=2 to 350, $N$ increment by 15 |

### Learning Bit-representations of Letters with the Pseudo-inverse Learning Rule

In the third experiment, we measure the recall accuracy of the model modified with pseudo-inverse learning on letter representations. We tested on 156 node representations of letters as demonstrated in Figure 2. The network was tested from $p = 2$ to $p = 26$ to determine whether the pseudo-inverse learned network could handle structured, and consequently more correlated, patterns.

## Results and Discussion

The model proposed by Lawrence, Trappenberg and Fine has been shown to be capable of recalling randomly generated patterns, such as 20 patterns using 1000-node patterns, without error [**5**]. However, this model could not recall the structured letter representations as in the first experiment. Only the first two patterns were recalled in the correct order. Because the patterns were structured, this failure was

likely due to the high correlation of bits between the patterns in the sequence.

The storage capacity of the model which uses Hebbian learning was $\alpha \approx .011$. We obtain this value by linear regression of the maximum number of patterns recalled with 100% accuracy over the number of nodes, using MATLAB.

The accuracy at which patterns were recalled for a given number of nodes and number of patterns using the Lawrence model can be found in Figure 3. The pseudo-inverse learning results are found in Figure 4. There is a dramatic increase in the storage capacity as compared to Figure 3. Recalling that storage capacity is the ratio of the maximum number of patterns correctly recalled divided by the number of nodes in each pattern, we can see that for a reasonably high accuracy, such as 95%, the maximum number of patterns stored in Figure 4 is at least as high as, and generally considerably higher than, the maximum number of patterns stored in Figure 3, for a given *N*.

We now have $\alpha \approx .7$. Compared to the 20 patterns stored using 1000 nodes in [**5**], we can now store 20 randomly generated patterns with as few as 30 nodes. The improvement on this coupled network is comparable to the improvement in using pseudo-inverse learning over Hebbian learning in auto-associative networks. The problem of singular weight matrices did not arise after *N* exceeded approximately 10 nodes, so the method is even more promising.

$$N_{hebb} = \frac{1}{\alpha_{hebb}} = \frac{1}{.011} \approx 90.909 \tag{15}$$

$$N_{pseudo} = \frac{1}{\alpha_{pseudo}} = \frac{1}{.7} \approx 1.42857 \tag{16}$$

$$\frac{N_{pseudo}}{N_{hebb}} = \frac{1.42857}{90.909} = 0.0157143 \tag{17}$$

$$\left(\frac{N_{pseudo}}{N_{hebb}}\right)^2 = 0.000246939 \tag{18}$$

Increasing the storage capacity also improves computational efficiency. Most calculations involved in the process of recalling a sequence of patterns are the multiplications involved in Equations (1) and (3), which are of order $N^2$. The number of nodes required to recall *p* patterns with the Hebbian learning model and the pseudo-inverse learning model are given by $p*N_{hebb}$ and $p*N_{pseudo}$, respectively. $p*N_{hebb}$ is given by Equation (15) and $p*N_{pseudo}$ is given by Equation (16). Using Equation (17), the ratio of calculations to recall *p* patterns using

pseudo-inverse learning to the calculations to recall $p$ patterns using Hebbian learning is given by Equation (18). Thus, the calculations are reduced to approximately 0.0247% of what was originally required, or there is an improvement factor of approximately $10^3$.

The pseudo-inverse learned network is able to recall all 26 letter representation patterns without error. See Figure 5 to observe the matched pattern for a given time.

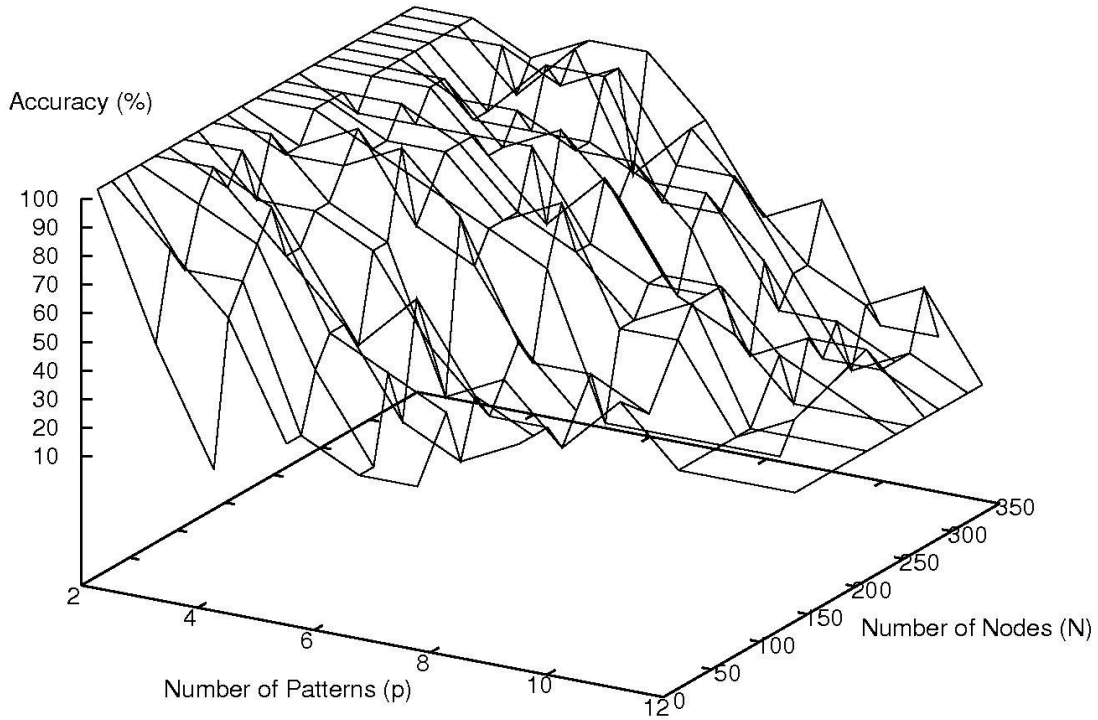



**Figure 3:** Accuracy of pattern recall against the number of nodes and number of patterns with Hebbian learning.

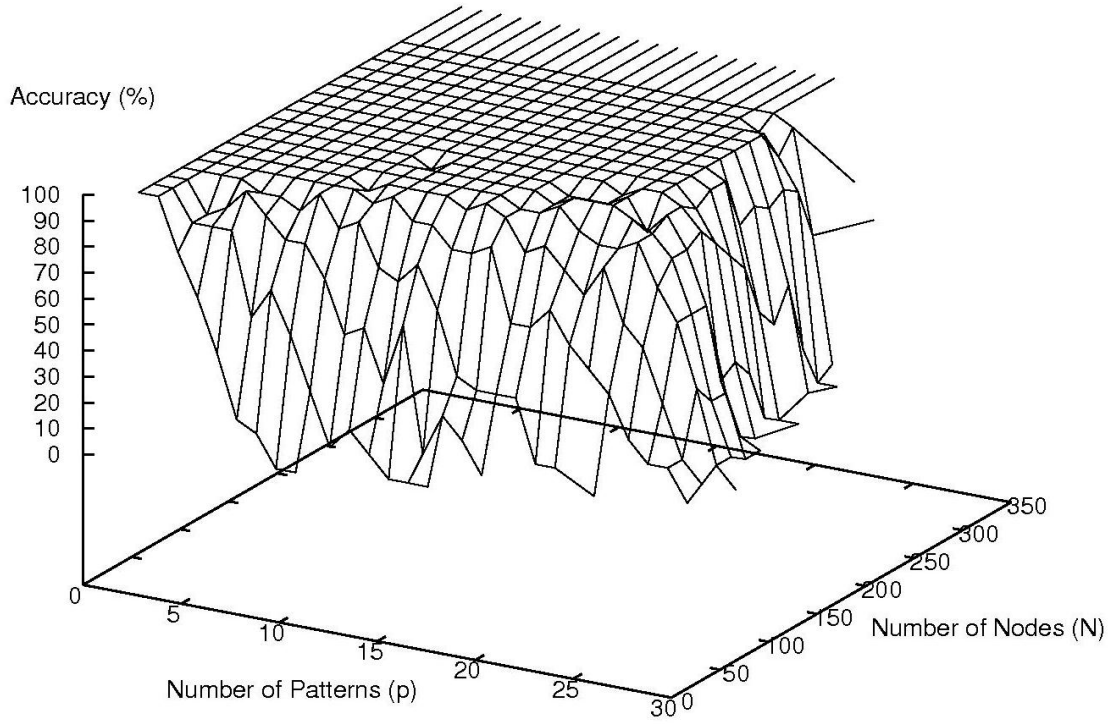



**Figure 4:** Accuracy of pattern recall against the number of nodes and number of patterns with pseudo-inverse learning.

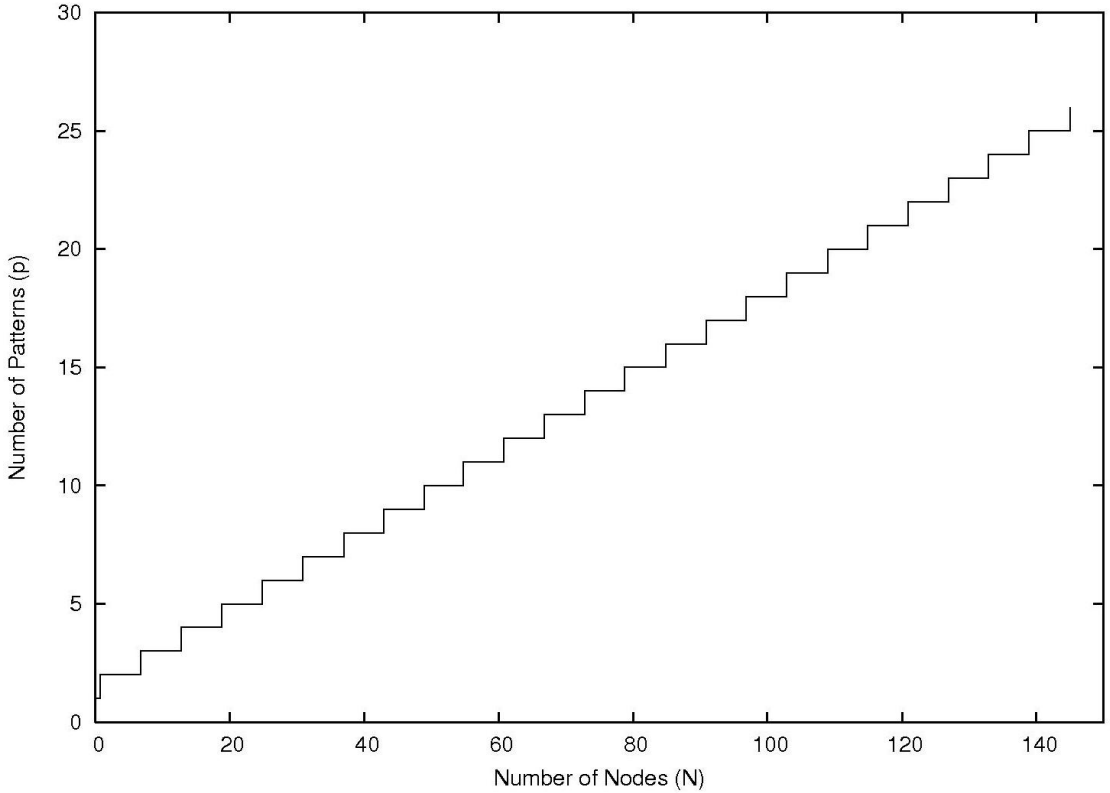



**Figure 5:** Number of the pattern in the letter sequence matched at a given time for pseudo-inverse learning.

## Conclusions and Future Work

As with the improvement of approximately 700% in terms of storage capacity for auto-associative memory recall using pseudo-inverse learning, this learning rule has also proven very useful in auto- and hetero-associative recall of simple sequences. The computational efficiency has an improvement factor of $10^3$. This makes the model more feasible for real-world applications.

Additionally, the modified network can now be used on a specific sequence of structured patterns: bit representations of letters. While this sequence was the only structured sequence to be experimented on, it is highly probable that there are other correlated patterns that can be handled by the new model network. Possible future work includes testing longer sequences of similarly correlated patterns and determining the relationship between the amount of correlation and the storage capacity.

One possible future study mentioned in [**5**] was examining networks to model complex sequences. With the lower storage capacity in the Lawrence-Trappenberg-Fine model, building on this network to accomplish that task appeared as though it would be very difficult. However, now that this modified model can handle much longer sequences, it is likely to be a good candidate for work in the direction of applying neural networks models to real-world problems.

## References


1

S. Coombes and J. G. Taylor *The storage and stabilisation of patterns in a hopfield net*, 1995, <**Neural Network World, 5(2):133-150**>.

2

S. Grossberg *Some networks that can learn, remember and reproduce any number of complicated space-time patterns, II*, 1970, <**Studies in App. Math. 49 (2) (1970) 135-166.**>.

3

O. Jensen and J.E. Lisman *Theta/gamma networks with slow NMDA channels learn sequences and encode episodic memory: role of NMDA channels*, 1996, <**Learning and Memory 3 (1996) 264-278**>.

4

T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka *Stock market prediction system with modular neural networks*, 1990, <**Neural Networks, 1990 IJCNN International Joint Conference, 1-6 vol.1**>.

5

M. Lawrence, T. P. Trappenberg, A. Fine *Rapid learning and robust recall of long sequences in modular associator networks*, 2006, <**Neurocomputing 69 (2006) 634-641**>.

6

J.E. Lisman *Relating Hippocampal Circuitry to Function: Recall of Memory Sequences by Reciprocal Dentate-CA3 Interactions*, February, 1999, <**UNeuron, Vol. 22, 233-242, Cell Press**>.

7

J.E. Lisman, L.M. Talamini, and A. Raffone *Recall of memory sequences by interaction of the dentate and CA3: A revised model of the phase precession*, 2005, <**Neural Networks 18 (2005) 1191-1201.**>.

8

M.D. Odom, R. Sharda *A neural network model for bankruptcy prediction*, 1990 <**Networks, 1990, 1990 IJCNN International Joint Conference, 163-168 vol.2**>.

9

Thomas P. Trappenberg *Fundamental of computational neuroscience*, <**Oxford University Press, p181-192**>.


## Biography

Kate Patterson received a BSc degree from Saint Mary's University in Halifax, Canada in 2006. She is currently an NSERC postgraduate scholar working towards an MCS degree from Dalhousie University. Her research interests include information retrieval, algorithms, and graph theory.