Neural Processing Letters (2006) 23:27–45
DOI 10.1007/s11063-005-2838-x

A Neural Model for Context-dependent Sequence Learning

LUC BERTHOUZE* and ADRIAAN TIJSSELING
Neuroscience Research Institute, Tsukuba AIST Central 2, Umezono 1-1-1, Tsukuba 305-8568, Japan. e-mail: luc.berthouze@aist.go.jp

Abstract. A novel neural network model is described that implements context-dependent learning of complex sequences. The model utilises leaky integrate-and-fire neurons to extract timing information from its input and modifies its weights using a learning rule with synaptic noise. Learning and recall phases are seamlessly integrated so that the network can gradually shift from learning to predicting its input. Experimental results using data from the real-world problem domain demonstrate that the use of context has three important benefits: (a) it prevents catastrophic interference during learning of multiple overlapping sequences, (b) it enables the completion of sequences from missing or noisy patterns, and (c) it provides a mechanism to selectively explore the space of learned sequences during free recall.

Key words. contextual cueing, incidental learning, leaky integrate-and-fire neurons, recurrent neural network, sequence learning

1. Introduction

The importance of context in storing and recalling (sequential) information has been thoroughly studied both in the animal and in the human domain. The contextual priming of memories seems to be one means that insects use to organise their knowledge and to retrieve memories appropriately. Collett et al. [1], for example, showed that spatial contextual cues condition the recall of visuomotor associative memories in the honeybee. Experimentation with snails [2] showed long-term memory (LTM) to occur only when the animals were tested in the context in which they were trained. In humans, motor learning studies also provide evidence that temporal information can be stored for different contexts without carryover from one to another [3]. When a subject's arm was trained on overhand throws and then tested on underhand throws, for most individuals the overhand training did not carry over to the subsequent underhand throws [4]. Yet, the overhand training persisted through to subsequent overhand throws, readapting with repeated throws. Finally, various theories of hippocampal functioning have stressed the role of context in both learning and predicting sequences [5, 6].

From a computational viewpoint, context can be seen as a stimulus environment that is changing much slower than the stimuli being learned [7]. Its (relative) stability generates invariants which facilitate the detection and identification of events [8].

*Corresponding author.

Yet, existing neural models of sequence learning have mostly assumed context as a cumulative record of previous inputs [9, 10] used to determine the next pattern.

The purpose of this paper is to describe a context-dependent sequence learning mechanism and to analyse it in terms of its ability to handle multiple, complex and possibly overlapping sequences, a problem that existing sequence learning models do not address properly.

2. Model

2.1. OUTLINE

The proposed architecture is shown in Figure 1. The core component of the model is a central module consisting of an array of independent columns of leaky integrate-and-fire neurons. Each column is characterized by the time constant of its neurons (see Section 2.2). Incoming input patterns are sequentially presented to an input module with no processing function. This input module has fixed, unit weight, one-to-one connections to each column of the central module. Context

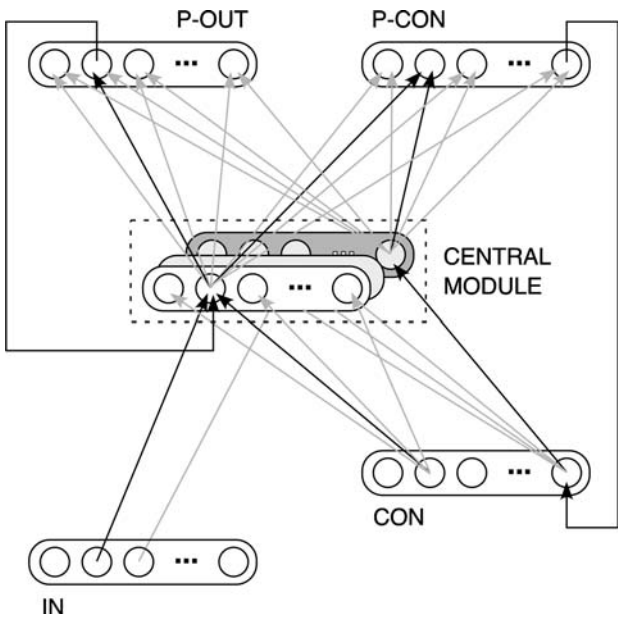

Figure 1. Schematic of the proposed architecture. The input module (IN) is a placeholder with no processing function. IN connects to each column of the central module via one-to-one connections. Each column in the central module is of the same size as IN. There are no lateral connections between each column of the central module. P-OUT is the output module. The same size as IN, it predicts the next input to the network by combining information from all neurons in the central module. P-OUT feeds back to each column of the central module via one-to-one connections. Context is processed in a similar fashion. Context module CON connects to all columns of the central modules via all-to-all connections. It receives both external contextual information and feedback information from the predicted context module (P-CON) of same size. The size of CON and IN are not necessarily identical.

patterns, instead, are fed to a context module of arbitrary size with fixed, random weight, all-to-all connections to each column of the central module. Each column has modifiable all-to-all connections to both the output module and the predicted context module. The weights are updated when the current pattern has been processed by the central module and the next one is presented to the input module. The learning rule (Section 2.4) aims to minimize the error between the current output and the next input. An efferent copy of the predicted context is fed back to the context module through fixed, unit weight, one-to-one connections. Finally, a set of fixed, unit weight, one-to-one connections link the output module and each column of the central module. This efferent copy is integrated to incoming activations from both input and context modules using a function described in Section 2.3.

2.2. NEURON MODEL

The design of the neurons in the central module is based on the Spike Accumulation Model by Shigematsu *et al.* [11] Each neuron has an accumulated potential u that is calculated using a leaky integration of its input I:

$$u(t) = I(t) + \tau \cdot v(t-1) \tag{1}$$

where $v(t)$ is the internal potential of the neuron at time t and τ is the decay rate of that internal potential. In conjunction with the firing threshold (see Equation 3), this last parameter determines the time to first pulse by controlling the minimum activation needed for the neuron to accumulate and fire.

The internal potential v is defined as:

$$v(t) = u(t) - \rho \cdot o(t) \tag{2}$$

where $o(t)$ is the output of the neuron at time t and ρ is the subtraction constant of the internal potential. This parameter controls the refractory period [12] of the neuron after firing, i.e. the time during which it cannot be excited (absolute refractoriness) or only with a much larger input (relative refractoriness).

In the original model, the output function o was defined by a Heaviside of the accumulated potential. In this model, however, it is given by:

$$o(t) = f(u(t)) = \frac{1}{1 + e^{-c(u(t)-T)}} \tag{3}$$

where c is the efficacy constant of the accumulated potential $u(t)$ and T is the firing threshold. From a neuronal dynamics viewpoint, this use of a sigmoid function as output function is justified by studies of the average of large numbers of neurons [13]. Indeed, the effect of averaging is to smear the all-or-nothing threshold potential of individual neurons into a sigmoid relationship between local mean dendritic potential and local mean firing rate [14]. Another justification can be

found at the dendritic level. Mel [15], for example, discussed how each dendritic compartment applies expansive nonlinearity to its synaptic input so that weak inputs are made weaker, and/or strong inputs are made stronger. With a simplified version of this model [16], it was shown that the use of nonlinear output pulses improved the robustness of the network to noise in the input sequence, and reduced its sensitivity to the choice of parameters.

By controlling for the time to first response, duration of the response, and time till the neuron settles back to quiescence (i.e. the time-constant of the neuron), τ, ρ and T also determine the network's ability to encode the timing information contained in an incoming sequence. A similar computing principle was used by Reiss and Taylor [17].

2.3. ACTIVATION PROPAGATION TO THE CENTRAL MODULE

Neurons in the central module receive a combination of the output of neurons from the input and context modules, and the output of neurons from the output module. Such recurrent connection is not novel. In Jordan [18], for example, the hidden layer is fed with the average of the outputs of both input and context neurons. In this network, the input I to neuron i in the central module is given by:

$$I_i = \frac{|o_i^I - o_i^O| \cdot o_i^I + o_i^O + o_i^C}{2.0 + |o_i^I - o_i^O|} \tag{4}$$

where o_i^I, o_i^O are the output values of the ith neuron from, respectively, the input and output modules, and o_i^C is the contribution of the context module to the ith neuron of the central module (see Equation 5). Unlike the function used in Jordan's networks, this function does not simply normalize the combined input, it also implements novelty filtering. When a sequence is recognized, the difference between o_i^I and o_i^O is small and the network shifts to recall (or predictive mode). This recall phase reinforces learned sequences (consolidation) and prevents overtraining.

Since the context module connects to the central module through fixed, randomly distributed, weights w^C in the range $[-1.0, 1.0]$, its contribution o_i^C to neuron i of the central module is given by:

$$o_i^C = f\left(\sum_j w_{ji}^C a_j^C\right) \tag{5}$$

where f is a simple sigmoid function

$$f(x) = \frac{1.0}{1.0 + e^{-x}} \tag{6}$$

and a_j^C is the activation of neuron j in the context module. This activation is computed from the input context pattern and the output of neurons from the predicted context module through one-to-one connections such that:

$$a_j^C = \frac{|I_j^C - o_j^{PC}| \cdot I_j^C + o_j^{PC}}{1.0 + |I_j^C - o_j^{PC}|} \tag{7}$$

where o_j^{PC} is the output of neuron j in the predicted context module, and I_j^C is the jth component of the input context pattern. As in the central module, this recurrent connection enables the network to recall a sequence when contextual cues are incomplete or noisy.

2.4. LEARNING RULE

Neurons in the output (and predicted context) module perform population coding of all neuron outputs in the central module. The output of neuron i is given by:

$$o_i = \frac{1}{1 + e^{-a_i}} \tag{8}$$

with

$$a_i = \sum_k \sum_j w_{ij}^k \cdot o_j^k \tag{9}$$

where o_j^k represents the output of neuron j in column k.

The weights w from the columns in the central module to the output module are updated according to an error-correction rule with synaptic noise given by:

$$\Delta w_{ij} = \alpha \cdot (\text{in}_i - \text{out}_i) \cdot o_j + \eta \tag{10}$$

where in_i is the output of neuron i in the input module (i.e. the next pattern in the sequence), out_i is the output of neuron i in the output module, o_j is the output of neuron j in the central module, α is the learning rate, and η is a noise parameter[1] of random value in the uniformly distributed range $[-\eta_{\min}, \eta_{\max}]$. This learning rule forces the weights to adapt to the difference between the actual output and the feedback information from the input module, multiplied by the activation distribution in the column. Since the input module holds the next pattern in the sequence, this rule minimizes the error between actual and predicted pattern at time $t+1$.

[1] In artificial neural networks, the application of synaptic noise to the learning process has been shown to improve convergence time and generalization performance to longer sequences [19]. In neurobiology, synaptic noise has been hypothesized to constrain the coding accuracy in neural structures [20] and enhance signal detection [21] among other roles.

3. Simulations

3.1. TRAINING PROCEDURE

A balanced learning of all sequences reduces the possibility of catastrophic interferences. In the following simulations, the following training procedure was used. Each sequence (and the corresponding context) of a permuted subset (initially, one sequence only) of sequences was trained for 10 iterations (an epoch). Each epoch was followed by a recall of each sequence of the subset. If the recall was successful, a new sequence was added to the subset (a block). Otherwise, an additional epoch of learning was carried out. A simulation was ended when all sequences were correctly recalled or when the number of epochs exceeded a predefined criterion in which case the network was said to have failed the learning task. This training procedure was also used by Dominey [9]. Retraining all sequences prevent instabilities that emerge when training a specific sequence shifts the weights excessively. In such case, the network finds a local minimum that solves the current sequence, but not all sequences. Repeatedly training all sequences forces the network to search for a global minimum. For complex sequences, Dominey [9] manually changed the learning rates to prevent deadlocks in the system. In this model, however, such step was not necessary since the network autonomously shifts to consolidation when a sequence is recognized, thus preventing overtraining.

3.2. MEMORY CAPACITY

Memory capacity was tested on a set of single sequences of 16-bit patterns. These patterns were constructed by binarizing random values from an uniform distribution in the interval [0, 1] with a threshold of 0.3. The context inputs were set to zero for all sequences to allow for comparisons with an Elman network of similar complexity [10]. An Elman network is a three-layered feed-forward neural network that uses backpropagation-through-time learning and is widely used for sequence learning.

Unless specified otherwise, the same parameter values were used for all simulations. The values are provided in Appendix.

3.2.1. *Sequence length*

Sequence length was varied from an initial 5 patterns to a maximum of 75 patterns in steps of 5. For each sequence length, 10 different sets of random patterns were created and both networks were trained on each sequence for 10 runs. The number of columns in the central module was set to 8 (see Section 3.2.2) with 16 neurons (i.e. the pattern size) in each column. Sixteen hidden neurons were used in the hidden layer of the Elman network.

In both networks, the relation between the number of epochs and the length of the sequence was found to be exponential (Figure 2). However, whereas the

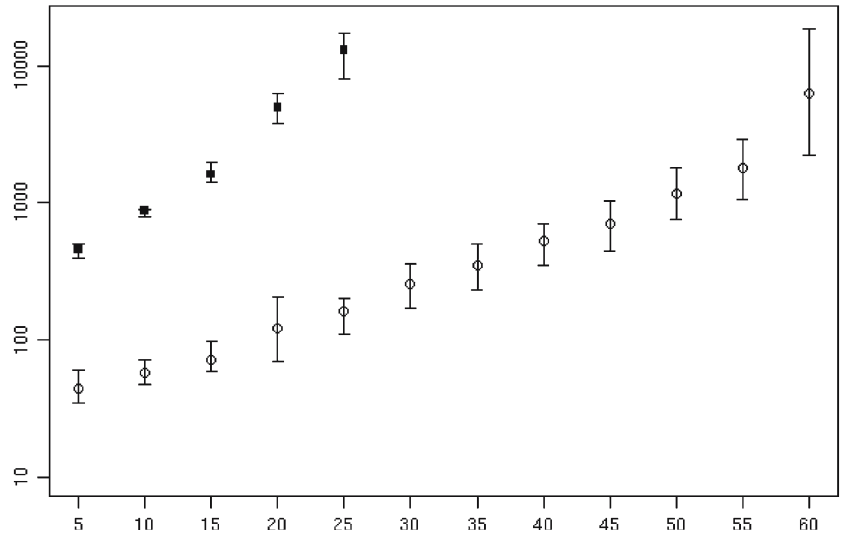

Figure 2. Effect of sequence length (horizontal axis) on the number of epochs (vertical axis, logarithmic scale) required for correct recall for both the proposed model (circles) and the Elman network (solid squares).

proposed network learned sequences up to a length of 60 patterns, the Elman network failed to learn sequences longer than 25 patterns (at least within the pre-set limit of 100,000 epochs), and otherwise required 33 times more epochs on average.

3.2.2. *Number of columns in the central module*

The relationship between the number of columns in the central module and recall performance was investigated by varying the number of columns from 1 to 50 for a given sequence of length 10. As shown by Figure 3, the proposed network showed a performance profile typical of overlearning followed by underlearning. The recall error sharply decreased initially until a maximum recall performance was reached (8-column configuration). With additional columns, however, performance became unstable.

While an overly large number of degrees of freedom might not allow the network to escape from local minima, another factor to consider was the complexity of the sequence itself, in particular, the number of degrees of freedom involved in each transition of the sequence. In a separate simulation, 10 different sequences of length 5 and pattern size 4 were used to investigate the correlation between the number of epochs required for correct recall and the complexity of the sequence, including the occurrence of repeated patterns or subsequences. The complexity of the sequence was evaluated in terms of the spread of its histogram of activities, and inter-pattern distances as evaluated by two different Minkowski distance functions. The number of epochs required to correctly recall each of the sequences

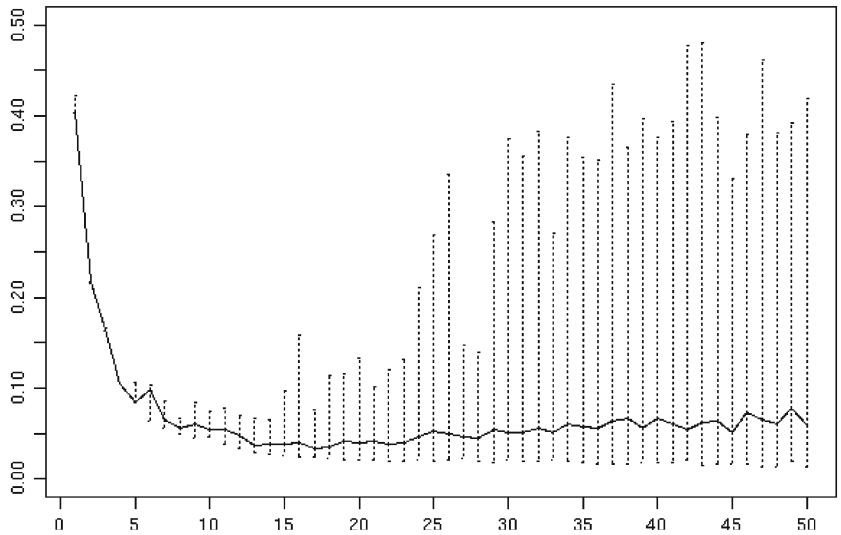

Figure 3. Effect of the number of columns in the central module (horizontal axis) on recall error (vertical axis). The error is computed as the square sum of errors between recalled sequence and correct sequence, averaged over the number of patterns in the sequence.

ranged from 932 to 30797, with a clear inverse correlation ($r < -0.83$) between inter-distance pattern and number of epochs (see Figure 4).

3.2.3. *Pattern size*

Using sequences of length 5–50, pattern sizes were varied from 4 to 20 in steps of 2. As expected from the discussion in Section 3.2.2, sequences of smaller patterns required more epochs because of the smaller inter-pattern distance. Still, the proposed network largely outperformed Elman's network, both in terms of the number of epochs needed to learn each sequence, and in terms of the length of sequences that could be learned (Figure 5).

The difficulty of Elman networks to learn long sequences might stem from the fact that, unlike the central module in the proposed network, the hidden layer only carries information relevant to the previous time step. This hypothesis was tested by comparing the performance of both networks in learning sequences in which one pattern at least was repeated.

3.2.4. *Encoding of timing information*

Given a sequence of length 5 and pattern size 16 (represented, for simplicity, by the string ABCDE), the capacity of both networks to encode timing information was analyzed by systematically incrementing the number of consecutive presentations of each pattern in the sequence: ABCDE, AABCDE, AABBCDE,

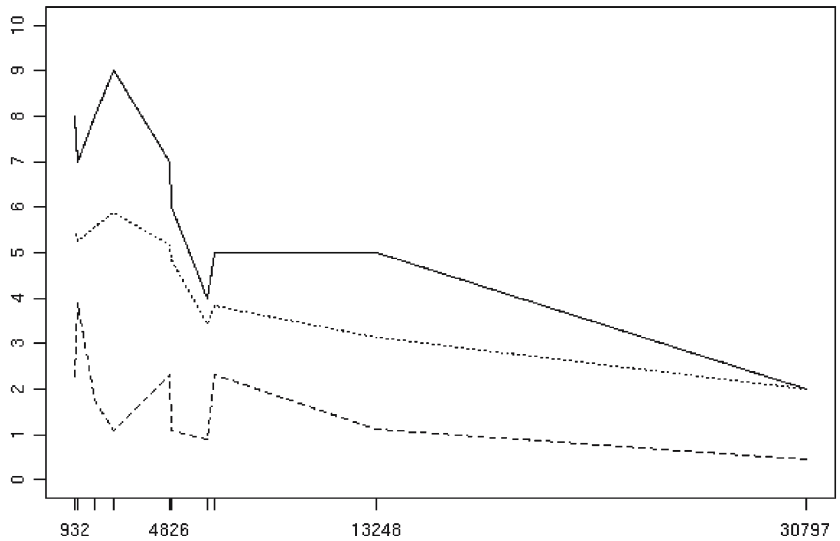

Figure 4. Inverse correlation between the number of epochs required to train a sequence (horizontal axis) and the complexity of the sequence (vertical axis), as approximated by the distance between consecutive patterns (using Minkowski functions of order 1 and 2) and the spread of activities of the patterns in the sequence.

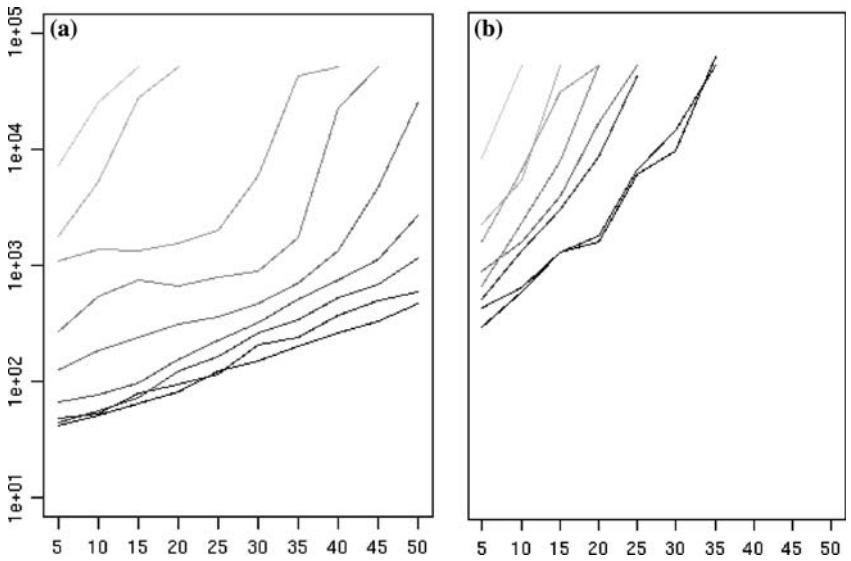

Figure 5. Effect of pattern size (horizontal axis) on the number of epochs (vertical axis, logarithmic scale) required to learn the sequence when the number of columns (proposed model, panel a) and hidden layer units (Elman network, panel b) varied from 4 (light gray) to 20 (black) in steps of 2.

..., AAAAABBBBBCCCCCDDDDDEEEEE. Sequences for which the number of epochs required for successful recall exceeded a preset criterion of 20,000 for 3 of the 5 sequences were excluded. The simulation consisted of 5 runs of 5 different random pattern sequences. As shown by Figure 6, the proposed network performed robustly, with an exponential relationship between the number of epochs required for successful recall and the complexity of the timing information. The Elman network, on the other hand, failed to learn any sequence that had at least one repetition, thus confirming our hypothesis of Section 3.2.3.

Since repeating patterns resulted in the lengthening of the sequence, the contribution of the repetitions to the overall complexity of the task was assessed by comparing the slope of the linear regression in the data of Section 3.2.1 and that of the data presented in this section (both using a logarithmic scale for the epoch numbers). A factor 2 (roughly) was observed (0.186 versus 0.081) for the repetition data.

3.3. LEARNING MULTIPLE, OVERLAPPING, SEQUENCES

3.3.1. *Problem*

To test performance with overlapping and branching sequences, a training set was derived from a real-world problem. In the Playstation^{TM} game Tekken^{TM} by Namco^{TM}, fighting characters are controlled by a human player using a console. Connected sequences of wrestling moves can be obtained via a complex sequence of button presses. Two distinct single moves (which will be referred to as A and F hereafter) provide a partial context to distinguish between otherwise overlapping

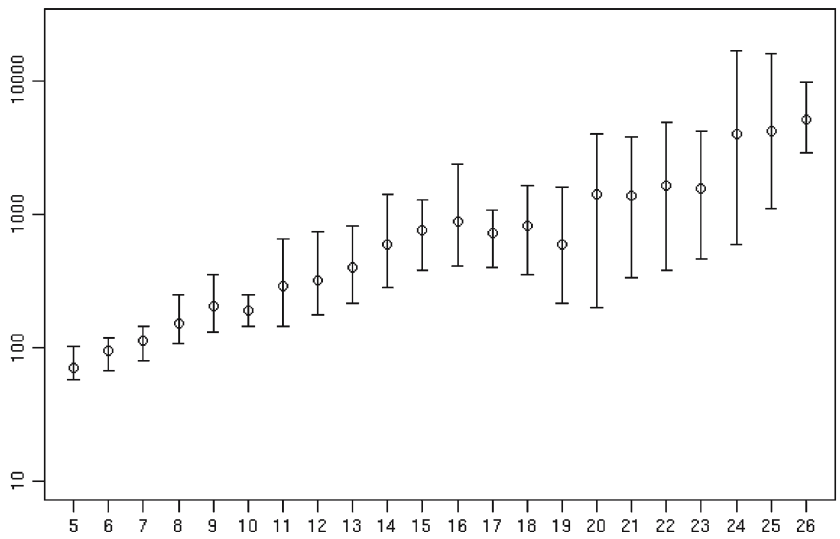

Figure 6. Effect of timing complexity (the horizontal axis denotes the number of repetition) on the number of epochs (vertical axis, logarithmic scale) required to learn the sequence.

sequences. The published multi-throw chart (see Figure 7) shows 10 possible complete sequences, 6 in context A and 4 in context F, with any subsequence constituting a valid move provided order is preserved.

Each multi-throw was converted into a 16-bit vector with the first 4 bits of the vector encoding the direction of movement, and the last 12 bits encoding a succession of up to 3 combinations of button presses. The context for each sequence was also constructed as a 16-bit string indicating for each sequence its starter and index.

3.3.2. *Learning of the Tekken sequences*

The complexity of the task was assessed by first learning the sequences without context module. The network failed to learn a second sequence, displaying a clear catastrophic interference effect [22]. Switching between sequences resulted in small, highly correlated ($r > 0.9$), shifts in weight space (Figure 8) that prevented a suitable distribution of the representations.

With context, however, all sequences were learned (Figure 9) with an expected increase in the average number of epochs required for correct recall at each new sequence. On the other hand, the number of iterations required to train each sequence before consolidation (Section 2.3) decreased suggesting that once the structure of the input domain had been acquired, the network merely fine-tuned the weights to generate correct recalls for all sequences.

As a generalization of the finding of Section 3.2.2, the number of epochs required to train a set of sequences was found to be correlated to the overlap

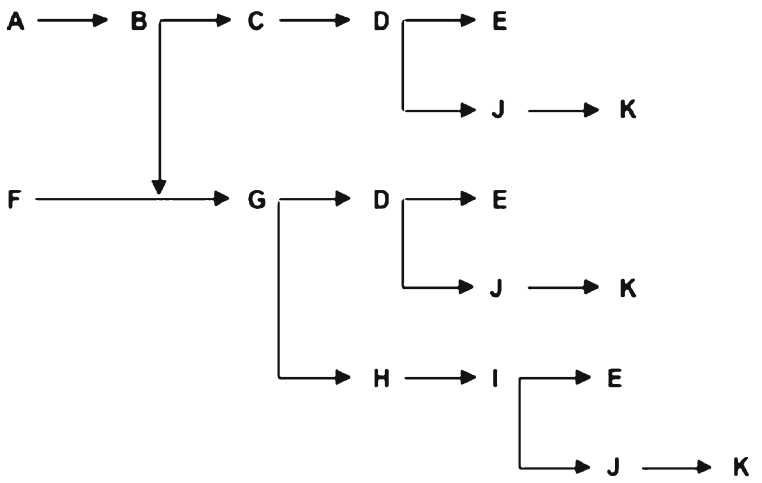

Figure 7. Multi-throw chart for the character King of Namco™'s Tekken™ game, obtained from an internet published documentation. Each letter corresponds to a multithrow, the combination of up to 3 button presses. Sequences of multithrows produce complex wrestling sequences. These sequences can only start with either of two starters A and F.

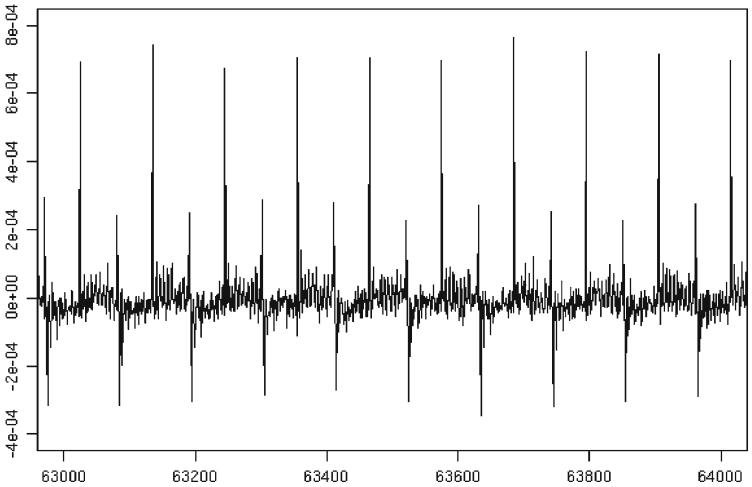

Figure 8. Changes in weights (vertical axis) of connections between a single neuron in the central module to the output layer during learning without context. The graph focuses on the low-amplitude oscillatory regime that follows an initial phase of large changes. In the oscillatory regime, the network keeps switching between a configuration that produces a correct recall of the first sequence and one that produces a correct recall of the second sequence.

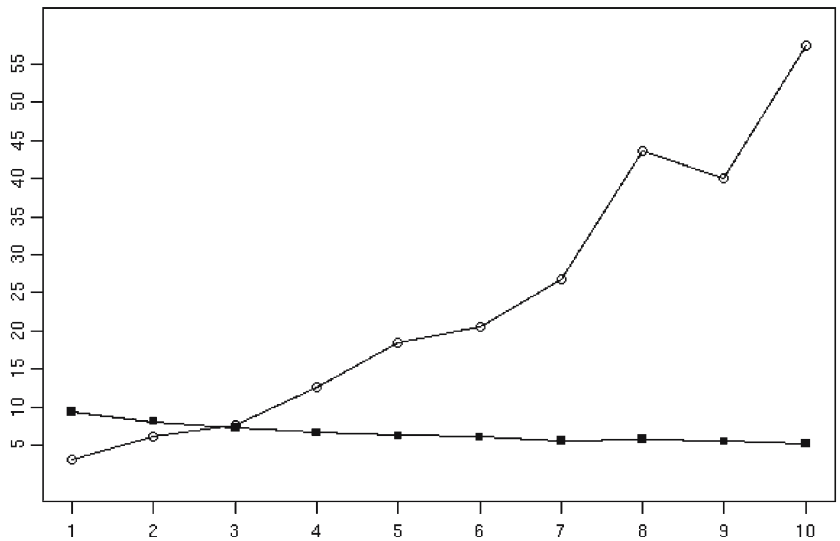

Figure 9. Average number of epochs (solid squares) for each sequence (horizontal axis) over 25 successful runs during which sequences were presented in permuted order. The circles denote the average number of iterations required to train each sequence (horizontal axis). The maximum number of epochs was set to 5000.

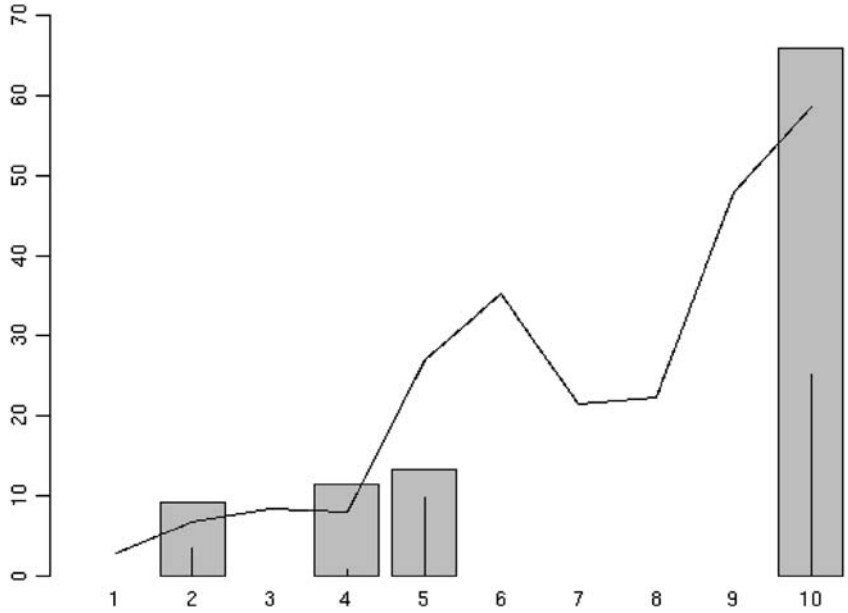

Figure 10. Effect of sequence overlap (sequences 1 to 6 and 7 to 10 have increasing overlap) on the average number of epochs (solid line) for each sequence (horizontal axis) when sequences are presented in non-permuted order. The bars and sticks denote the range (factor 10) and variance (factor 100) respectively of weight changes when sequences 2,4,5 and 10 are added. Learning the 4-th sequence required only a few epochs with a low variance of weight changes, indicating that memory for this sequence fitted well into the existing weight configuration. Adding the 5-th and 10-th sequences, however, required far more epochs, with an increased variance in weight changes. The large range of weight changes observed when adding the last sequence indicate a reconfiguration of the weight space.

between sequences. When presented in the order shown in Figure 7, each new 'A'-sequence required an increasing number of epochs, in particular the last 2 'A'-sequences which not only share a long partial sequence, but also introduce a new branch at move 'G'. At peak overlap (when the last two 'F'-sequences were introduced), the number of epochs required to train the whole set was significantly larger. A strong increase in epochs might also correspond to a reconfiguration of the weight space to accommodate a new sequence (the learning of specific transitions provided evidence for large weight rearrangements, as shown by Figure 10 for example), which raises the issue of memory structure.

3.3.3. *Structure of sequence representations*

To test the hypothesis that the network constructs a similarity structure between overlapping sequences of the training set, three additional simulations were realized. In the first one, an alternative training method was used to quantify whether learning complexity covaried with similarity structure complexity. A richer similarity structure was created by adding all possible sub-sequences (e.g., AB, ABC,

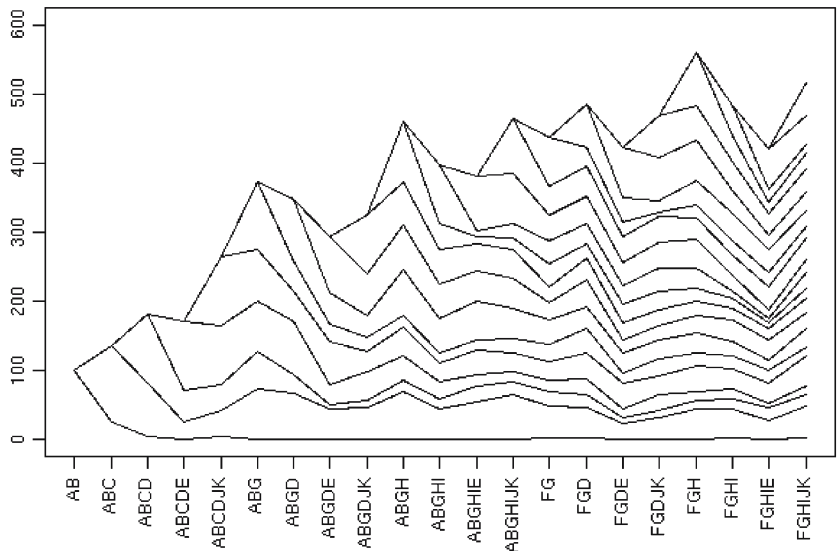

Figure 11. Stacked percentile contribution (vertical axis) of each sequence to the number of epochs required to train all sequences up to the current one (horizontal axis). When adding sequence ABCDE, for example, no retraining of AB was required while the retraining of ABC and ABCD took roughly 25% each of the total number of epochs required to correctly recall all sequences from AB to ABCDE.

ABCD) to the sequences of the original training set, for a total of 21 sequences. A linear sequence presentation (starting from AB and ending with FGHIJK) was used to avoid any order-related artifact in performance. Each set of sequences was repeatedly trained until each one of them was correctly recalled. The occurrence of a reconfiguration of the weights was determined by verifying which sequence needed retraining after presentation of a new sequence. As shown by Figure 11, a retraining phase of previous sequences only occurred when a new sequence introduced a branching point. Adding ABCDE to a set comprising ABCD, for example, mostly involved the training of this new sequence and the consolidation of previous sequences. On the other hand, the addition of ABCDJ, which introduces a branching point D{EJ}, required significantly more training. This observation was consistent across training blocks. The addition of ABG, for example, required the retraining of all sequences but AB. Further additions of sequences with alternative endings only required minor retraining of overlapping sequences. When a new pattern was introduced, as with H in ABGH, additional epochs were required to incorporate the new pattern as well as the extra subsequence GH. In addition, AB required more training trials with the introduction of a new starter F in FG. These results thus suggest that the learning of sequences proceeds by extracting the underlying similarity structure, defined by overlapping and branching parts.

In a second simulation, the prediction that introducing a sequence that did not match the similarity structure already acquired by the network would result in more epochs, was systematically tested by considering three conditions: addition

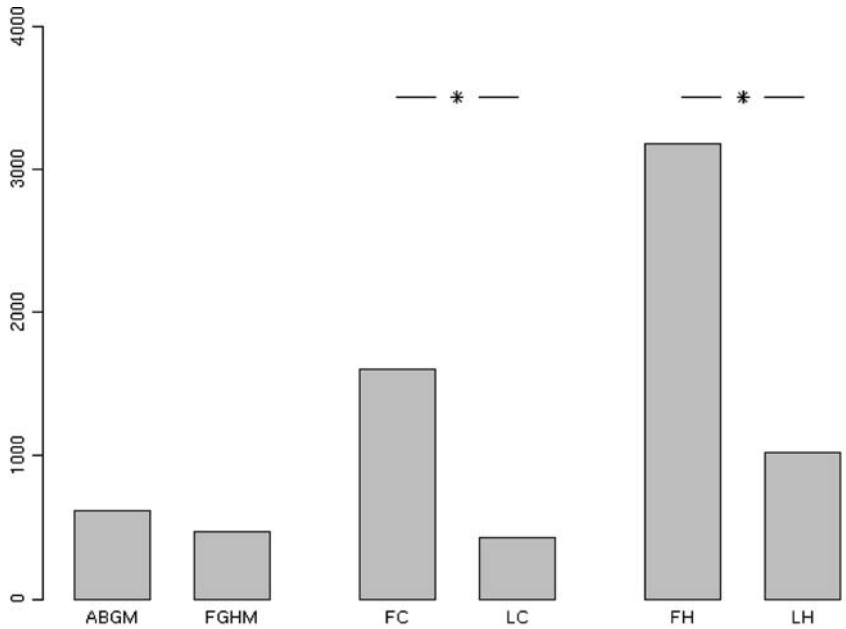

Figure 12. Main effects of learning a new pattern M (bars ABGM and FGHM), a transition between previously unconnected patterns (bar FC), and a shortcut between patterns of a same sequence (bar FH). Stars denote statistical significance ($p < 0.1$) in t-tests with control conditions LC and LH. Statistical significance ($p < 0.05$) was reached when comparing either FC or FH with ABGM and FGHM.

of a new ending move, addition of a new context, and addition of a new transition connecting previously unlinked moves (shortcut). As a control condition, two new sequences were constructed by adding a new ending move M to sequences ABGH and FGH. A new sequence FC connected two moves that had not been used in the same sequence. To verify the effect of adding sequence FC to existing F-sequences, a sequence LC was also added linking C with a new starter L. Finally, a new sequence FH (shortcut for FGH) was compared with LH which involves a new context. To avoid any weight-related artifact in performance, the same weight configuration (obtained after successful training of the 21 sequences) was used to train each of the 6 new sequences. As shown by Figure 12, introducing a new terminator M did not have a significant effect on the number of epochs required to train the final set of sequences. A significant effect, however, was found when learning transitions (FH,FC) that broke the similarity structure. Shortcut FH, for example, required nearly 2200 more epochs than LH to be learned. This shortcut, which amounts to adding a new branching point at F, resulted in the retraining of FG and FGH, and to a lesser extent of all other sequences containing GH. Similarly, learning FC took significantly more epochs than learning LC ($p < 0.1$).

The third simulation involved the free recall of the memory structure after severing the feedback loop between predicted context module and context module. By

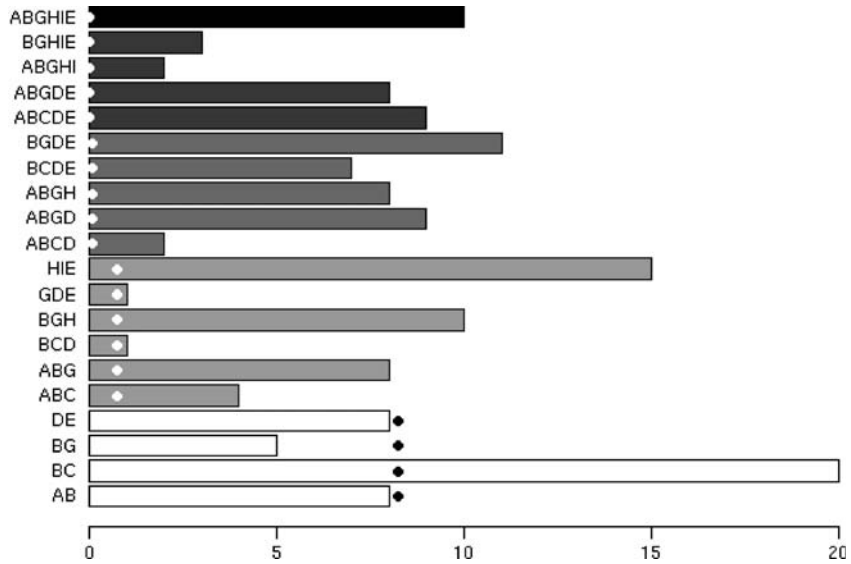

Figure 13. Number of occurrences (horizontal axis) for each sequence (partial and complete) during 1000 time-steps of free recall after cueing the network with starter A (partial context). The vertical bars denote the expected number of occurrences of sequences of length 2–6 in a string of long length randomly generated from an alphabet of 11 characters (A–K).

preventing the network from reconstructing the context, instabilities in the recall were expected that should reveal stable states in the memory structure by way of chaotic itinerancy [23]. A free recall was initiated with the network cued only with the starter of the first sequence and a partial context for that starter. The longest recognizable sequences were recorded over 1000 steps of recall, and their frequency of occurence was compared to the probability that they appear in a randomly generated sequence of long length. The probability that a word of length l occurred in a string randomly generated from an alphabet of n letter was computed as:

$$\left(\frac{1}{n}\right)^{l} \tag{11}$$

As shown by Figure 13, longer sequences, in particular, were found to occur at a rate significantly higher than that expected from a random draw. Sequence ABGHIE, for example, occurred 10 times when the probability that it occurred by chance was $5.6E-7$. The starting partial sequence ABG was found to occur relatively more often than ABC, thus reflecting its larger frequency of occurrence in the training set. Interestingly, sequences ending in JK were not recalled. Since JK was involved in three different sequences, its confusability was higher than that of HIE (the most recalled terminating sequence) for example. This hypothesis on the role of confusability in recall was tested by a second recall using starter F as partial context. Since G was involved in 8 different sequences, a very limited recall

was expected. Indeed, whereas G occurred most frequently and was sometimes followed by H and I, no full sequences were recalled.

4. Discussion

A novel context-dependent sequence learning neural network was proposed. Its learning process is driven by a dynamic adjustment of the learning rate, a process that provides the network with the ability to learn only when learning is necessary and revert to consolidation or rehearsal otherwise. It enables the network to address the stability-plasticity issue that occurs when new sequential information partially or completely overwrites memory of previously acquired sequences. The integration of context, in turn, prevents catastrophic interference between overlapping sequences. Context provides the network with slow-changing regularities that help it distinguish between various sequences based on when and where they occur. Without context, the network can only rely on individual patterns to determine what it has to predict next. Free recall when the context feedback loop was severed showed the network to exhibit chaotic itinerancy whereby the network jumped from one low-dimensional attractor (sequence representation) to the other because context could no longer stabilize network dynamics.

From a developmental viewpoint, an interesting feature of the model is its ability to dynamically reconfigure its memory structure when new information becomes available. This is made possible because the network does not match sequences with a stored sequences but rather implements a dynamical filter whereby incoming stimuli direct the network towards a portion of the state space which has been shaped by previous exposures to similar sequences. Simulations showed this mechanism to implement periodic-attractor rather than point-attractor dynamics, thus providing the substrate for a reconfigurable similarity structure. As such, the network could prove a useful component for a developing cognitive system that must deal with temporal information of increasing complexity.

Appendix

Table A. Parameter settings used in the model.

Central module		
τ^{max}	max decay rate	0.90
ρ^{max}	max subtraction constant	0.40
c	efficacy constant	20.0
T	firing threshold	1.0
Weight modification		
η^{max}	maximum noise	0.0001
η^{min}	minimum noise	−0.0001

The decay rate τ and the subtraction constant ρ of the neurons in the central module were evenly distributed in the interval $[\tau^{\max}, \tau^{\max}/(c+1)]$ and $[\rho^{\max}, \rho^{\max}/(c+1)]$ respectively, with c denoting the number of columns.

Acknowledgements

This study was funded by the Advanced and Innovational Research program in Life Sciences from the Ministry of Education, Culture, Sports, Science and Technology, the Japanese Government.

References

1. Collett, T., Fauria, K., Dale, K. and Baron, J.: Places and patterns – a study of context learning in honeybees. *Journal of Comparative Physiology A*, **181** (1997), 343–353.
2. Haney, J. and Lukowiak, K. Context learning and the effect of context on memory retrieval in lymnaea. *Learning and Memory*, **8**(1) (2001), 35–43.
3. Thach, W.: On the specific role of the cerebellum in motor learning and cognition: Clues from pet activation and lesion studies in man. *Behavioral and Brain Sciences*, **19**(3) (1996), 411–431.
4. Thach, W., Goodkin, H. and Keating, J.: The cerebellum and the adaptive coordination of movement. *Annual Review of Neuroscience*, **15** (1992), 403–442.
5. Hirsh, R.: The hippocampus and contextual retrieval of information from memory: A theory. *Behavioural Biology*, **12** (1974), 421–444.
6. Levy, W. and Wu, X.: The relationship of local context codes to sequence length memory capacity. *Network: Computation in Neural Systems*, **7** (1996), 371–394.
7. Wallenstein, G., Eichenbaum, H. and Hasselmo, M.: The hippocampus as an associator of discontiguous events. *Trends in Neuroscience*, **21** (1998), 317–323.
8. Chun, M. and Jiang, Y.: Contextual cueing: Implicit learning and memory of visual context guides spatial attention. *Cognitive Psychology*, **36** (1998), 28–71.
9. Dominey, P.: Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. *Biological Cybernetics*, **73** (1995), 265–274.
10. Elman, J.: Finding structure in time. *Cognitive Science*, **14** (1990), 179–211.
11. Shigematsu, Y., Ichikawa, M. and Matsumoto, G.: Reconstitution studies on brain computing with the neural network engineering. In: *Perception, Memory and Emotion: Frontiers in Neuroscience*, pp. 581–599. Elsevier Science Ltd., 1996.
12. Milton, J., Mundel, T. and van der Heiden, U.: Traveling activity waves. In: *The Handbook of Brain Theory and Neural Network*, pp. 994–997. MIT Press, Cambridge, MA., 1995.
13. Gerstner, W.: Spiking neurons. In: *Pulsed Neural Networks*, pp. 3–54. MIT Press, 1998.
14. Robinson, P., Rennie, C. and Wright, J.: Propagation and stability of waves of electrical activity in the cerebral cortex. *Physical Review E*, **56** (1997), 826–841.
15. Mel, B.: Why have dendrites? A computational perspective. In: *Dendrites*, pp. 271–289. Oxford University Press, Oxford, UK, 1999.
16. Tijsseling, A. and Berthouze, L.: A neural network for temporal sequential information. In: *Proceedings of the 8th International Conference on Neural Information Processing, Shanghai (China)*, pp. 1449–1454, 2001.
17. Reiss, M. and Taylor, J.: Storing temporal sequences. *Neural Networks*, **4** (1991), 773–787.

18. Jordan, M.: Attractor dynamics and parallelism in a connectionist sequential machine. In: *Proceedings of the Eight Annual Conference of the Cognitive Science Society*, pp. 513–546. Erlbaum: Hillsdale, NJ., 1986.
19. Jim, K., Giles, J. and Horne, B.: Synaptic noise in dynamically-driven recurrent neural networks: Convergence and generalization. Technical Report CS-TR-3322, University of Maryland, USA, 1994.
20. Zador, A.: Impact of synaptic unreliability on the information transmitted by spiking neurons. *Journal of Neurophysiology*, **79** (1998), 1230–1238.
21. Collins, J., Chow, C. and Imhoff, T.: Stochastic resonance without tuning. *Nature*, **376** (1995), 236–238.
22. French, R.: Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference. In: *Proceedings of the 16th Annual Cognitive Science Society Conference*, Vol. 5, pp. 207–220, 1994.
23. Tsuda, I.: Dynamic link of memory – chaotic memory map in nonequilibrium neural networks. *Neural Networks*, **5** (1992), 313–326.