Using Latent Attractors to Discern Temporal Order

Simona Doboli
Department of Computer Science
Hofstra University
Hempstead, NY 11549
E-mail: Simona.Doboli@hofstra.edu

Ali A. Minai
Complex Adaptive Systems Laboratory
ECECS Department
University of Cincinnati
Cincinnati, OH 45221
E-mail: Ali.Minai@uc.edu

***Abstract*— The paper presents a neural model for learning sequences of relevant patterns embedded in distractors. A contextual episode is a sequence of relevant patterns – always in the same order – intermixed with distractors. By repeated presentations of all contextual episodes, the model discovers for each episode the set of relevant patterns and their order. The problem is solved in two stages: (a) by eliminating distractors, and (b) by learning the order between relevant patterns. The model uses the concept of latent attractors - essential in creating different neural representations for same patterns in distinct episodes. No external teacher and only Hebbian type learning rules are used.**

I. Introduction

Most information available to a cognitive system occurs in time, and the temporal order of perceived informational entities often has great significance. A large body of work exists on neural networks that learn to predict or recognize sequences [27], [19], [17], [31], [24], [12], [18], [14], [15], [22], [16], [1], [28], [21], [13], [29], [4]. This work has addressed many of the important issues in sequence learning, including context-dependence, temporal structure, disambiguation, noise, etc. However, other issues remain open.

In the present paper, we focus on the problem of discerning a stable temporal order among sensory data when they occur with variable timing, and are interspersed with irrelevant distractors. Suppose a cognitive agent repeatedly performs action A which produces a well-defined sequence of consequent observations. However, on each trial, the relevant observations are separated by a variable number of irrelevant observations that have nothing to do with A but may be relevant at other times. The system's task is to detect and learn the relevant sequence of observations. A non-neural system can solve this problem relatively easily by performing off-line estimates of observation frequencies and temporal order. However, a connectionist system does not have this option; it must learn by adapting its synaptic weights based only on information available in real-time. Also, if it is to be a biologically plausible system, it cannot use supervised learning methods such as back-propagation [30], [26] or its temporal variants [31], [24]. Instead, it must rely on Hebbian long-term potentiation (LTP) [3], [20] and long-term depression (LTD) [25], [11] of synapses with its constraints of spatio-temporal locality. While there has been some work on learning sequences with Hebbian synapses [27], [18], [1], [22], [28], [21], most approaches for learning complex sequences have relied on supervised learning [19], [31], [24], [12], [14], [15], [2], [13], [4]. In this paper, we consider how the problem of extracting temporal order among observations can be solved by networks using purely Hebbian learning and no off-line storage or external teacher signals. This work uses the concept of latent attractors, which we introduced to model context-dependent spatial representations in the mammalian hippocampus [23], [9], [10], [7], and which we later applied to pattern recognition using sequential observations [5], [6], [8].

A key issue in solving the problem described above is that of *information latching*[2]: Retaining transiently available information for arbitrarily long durations to capture the non-Markovian dependencies of a sequence. Virtually, all neural models of sequence learning use tapped delay lines of fixed or adaptable [28] length, Randomly distributed delays between neurons, or decaying activity traces [27]. In all cases, the assumption is that recent information is more useful than temporally more remote information. This is often a reasonable assumption, but may not hold in many cognitive situations. For example, the context for most episodic experiences (e.g., conversations, meetings, etc.) is set at the beginning of the episode (e.g., recognizing the identity or mood of the interlocutor, identifying the composition of a group, recognizing a place, etc.) This initial recognition of context can be much more significant in determining later decisions or responses than events closer in time. Similarly, in the problem at hand, it is important for the system to discern and remember relevant observations, no matter how long ago they occurred, while discarding arbitrarily long sequences of irrelevant observations. Tapped-delay lines, random delays, and decaying activity are not sufficient to accomplish this [2]. In our previous work, we have termed this *long-term context* to distinguish it from the more usual, recency-based view of context in temporal information processing, and have proposed the use of attractor-based computation to address the issue [9], [10], [5], [6], [8], [7]. We use the same ideas in the present work.

The problem we consider is also relevant to sequential data mining, an area that has lately attracted much interest in the context of database and bioinformatics applications [32], [33]. However, almost all work in this area has used methods other than neural networks, since allowing off-line storage and inference greatly simplifies the problem. It is not clear at this point whether a neural approach would produce benefits in data mining applications, but it is instructive to

think of a cognitive system as being engaged in real-time, online mining of sequential data from experience, and insights from data mining may well prove useful in understanding cognitive systems.

II. Problem Description

We consider a neural system that receives time-varying input organized into sequences of *stimulus patterns*, each sequence corresponding to a *contextual episode*. The patterns in the sequences are drawn from two sets:

1) The set of *context patterns*, $\mathcal{C} = \{C^k\}$, with $k = 1, 2, ..., m$.
2) The set of *regular patterns*, $\mathcal{X} = \{X^k\}$, with $k = 1, 2, ..., n$.

Contextual episodes are drawn from an *episode list*, $E = \{ E_q \}$. Each contextual episode, E_q, has a *canonical sequence*, $\sigma^q = \sigma_0^q \ \sigma_1^q \ \sigma_2^q ... \sigma_M^q$, where $\sigma_k^q \in \mathcal{C}$ for $0 \leq k \leq L < M$, and $\sigma_k^q \in \mathcal{X}$ for $L+1 \leq k \leq M$. Thus, the first L stimulus patterns comprise a *context identifier*, and the remaining stimuli are observations. The canonical sequence is initially unknown to the system.

The system operation consists of successive *experiences*, R^T, $T = 1, 2, \ldots$. Each experience, R^T, corresponds to a specific contextual episode, $E(T) \in \mathcal{E}$. We assume that each contextual episode is experienced several times, interspersed with other contextual episodes. If $E(T) = E^q$, the *actual sequence* of patterns experienced by the system during experience R^T is: $S^T = s_0^T \ s_1^T \ ... \ s_{N_T}^T$, $N_T \geq M$, such that:

- $\sigma_k^q \in S^T$ for all $\sigma_k^q \in \sigma^q$, i.e., S^T includes the entire canonical sequence of the corresponding contextual episode.
- If $s_\alpha^T = \sigma_k^q$, $k = 0, 1, ..., M-1$, and $s_\beta^T = \sigma_{k+1}^q$, then $\beta > \alpha$, i.e., the patterns of σ^q always occur in the correct order within S^T.

Thus, the actual sequence during an experience is the canonical sequence for a specific episode with arbitrarily long sub-sequences of other patterns inserted in between. We term this the *embedding* of the canonical sequence in the actual sequence. We call patterns of the canonical sequence *relevant patterns* and the rest *distractors*. *Distractors* or *relevant* patterns in one contextual episode can be *relevant patterns* in other episodes. The only constraint is that relevant patterns in one contextual episode be unique.

As given, the system is required to perform two tasks during an experience:

1) Identify the contextual episode from context patterns.
2) Predict the canonical sequence for the episode.

In our previous work, we have shown how a latent attractor network can identify context in the presence of distractors [5], [6], [7], so in this paper we focus on the second task.

III. Model description

The architecture of the system is shown in Figure 1. It consists of several layers, with N_L denoting the number of neurons in layer L. Unless otherwise specified, each projection has random connectivity with a given probability of connection and all weights are chosen randomly with small initial values.

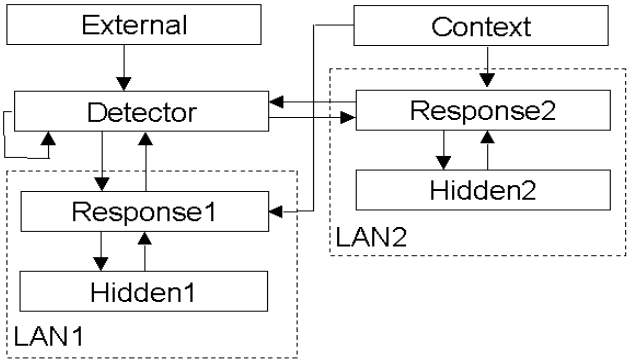

Fig. 1. Network architecture

The system's task is to learn sequences of relevant symbols — embedded in distractors — comprising a contextual episode. The problem can be divided into two sub-tasks:

- **Subtask I:** Learning to distinguish relevant patterns from distractors in each contextual episode.
- **Subtask II:** Learning the order of relevant patterns in each contextual episode.

The first subtask is accomplished by the External (E), Detector (D), and Context (C) layers, and the Latent Attractor Network 1 (LAN1) formed by the Response-1 (R1) and Hidden-1 (H1) layers. Latent Attractor Network 2 (LAN2) with Response-2 (R2) and Hidden-2 (H2) layers is activated only during the second subtask.

The role of the External (E) layer is to recognize input stimuli. To keep the model simple, we use a 1-of-N code for the inputs, so that each known pattern (without regard to relevance) activates a single, unique neuron in E. The E layer projects to the Detector (D) layer: Each E neuron is connected to its correspondent neuron in D layer. Coding in D layer is the same as in E layer, with one neuron per input pattern. The main difference lies in its recurrent connectivity — monosynaptic feedback from D itself, and a disynaptic projection through the R1 layer. The monosynaptic recurrent projection is a self-feedback of each D neuron onto itself. Activity in the D layer is competitive, so that the neuron with the largest excitation above a threshold fires at each time step.

The R1 and H1 layers together form a latent attractor network (LAN) [23], [10]. A 2-layer LAN is a disynaptic recurrent network with two layers: R and H. It has a number of attractors – each defined by a subset of neurons in both layers called the active set of that attractor – embedded in its recurrent connections via clipped Hebbian learning. A competitive $K - of - N$ firing rule activates K neurons, where K is smaller than the size of the active set of an attractor (N_A): $K < N_A$. The functionality of LAN is as follows: once an attractor becomes active - most K neurons are from an attractor's active set - the activity will remain inside that attractor as long as the external input onto R layer

is either uniformly distributed on all R neurons or it targets mostly neurons in the active set of the selected attractor. Active neurons, even though confined to the active set of an attractor, are determined by the current external input. In the model, the role of the two LANs is two fold: one, to represent the current contextual episode - by activating a distinct latent attractor in each episode, and second, to represent the external stimulus - relevant or distractor. By activating a different latent attractor in each episode, the same stimulus in different contextual episodes will have a different LAN representation. A simple attractor network could do either of the two tasks, but not both at the same time.

LAN1 receives projections from D as well as from the C layer. The role of the C layer is to recognize a change in contextual episode and to project this information to LAN1 and LAN2. C layer neurons are active only at the beginning of a contextual episode. Each contextual episode begins with a single context pattern — randomly chosen for each episode — activated in the C layer. The C-to-R1 and C-to-R2 projections are modified via LTP such that each C context pattern biases for activity mostly neurons in the active set of a distinct attractor. Once an episode is recognized and the correct pattern is activated in the C layer, the latent attractor chosen to represent the episode becomes active. After the initial recognition, C neurons are silent until a new contextual episode starts. This part of the system can be seen as a context-representation network as described in [5], [6], [7].

The system operates on two time-scales: A *slow time-scale*, indexed by t, that corresponds to the presentation of new external stimuli and update in the activity of E and D neurons; and a *fast time-scale*, indexed by τ, which embeds τ_f updates of R1, H1, R2 and H2 neurons in each cycle of the slow time-scale. Each episode/experience begins with $t = \tau = 1$, so the t value corresponding to a τ step can be obtained as: $t(\tau) = [\tau/\tau_f] + 1$. For notational economy, we denote the value $x(t(\tau))$ of a slow time-scale variable x as $x(t)$ when only the slow time-scale index is relevant.

A. Subtask I: Learning to Detect Relevant Patterns

The system first learns to accomplish Subtask I using repeated presentations (experiences) of all contextual episodes. Each experience starts with the appropriate context pattern in the C layer, followed by the relevant patterns intermixed with distractors in the E layer.

The first input in a contextual episode is the context pattern active in the C layer. The context pattern projects onto R1 layer and – due to its modified connections - it selects the attractor for the episode. The net input to neuron i in R1 is:

$$h_i^{R1}(\tau) = g_{DR1}(t) \sum_{i \in D} w_{ij}^{DR1} x_j^D(t) + g_{H1R1} \sum_{i \in H1} w_{ij}^{H1R1} x_j^{H1}(\tau - 1)$$

$$+ g_{CR1}(t) \sum_{i \in C} w_{ij}^{CR1} x_j^C(t) \quad (1)$$

where $g_{(.)}$ are the gains of each projection. The C layer input into R1 is active only at the beginning of a contextual episode: $g_{CR1}(t) = 0, \forall t > 1$ and the D layer input into R1 is active only for $t > 1$: $g_{DR1}(t) = 0, \ for \ t = 1$. Activity in the R1 and H1 layers is determined by a $K - of - N$ competitive firing rule, with the K most excited neurons in each layer fired every time step. H1-to-R1 projection maintains a uniform high bias onto the active set of the selected attractor.

Each subsequent input stimulus — a relevant or distractor pattern — fires its associated neuron in the E layer. The net input for a neuron i in the D layer is:

$$h_i^D(t) = g_{ED} \sum_{j \in E} w_{ij}^{ED} x_j^E(t) + g_{DD} \sum_{j \in D} w_{ij}^{DD} x_j^D(t-1)$$

$$+ g_{R1D} \sum_{j \in R1} w_{ij}^{R1D} x_j^{R1}(\tau - 1) \quad (2)$$

with $g_{(.)}$ denoting the gain of each connection. Firing in D layer is a threshold winner take all rule that fires the most active neuron if it exceeds a threshold, θ.

Initially, firing in D layer is determined solely by the dominant E layer projection. R1 input provides uniform bias from the active set of the selected attractor onto all D neurons. Before learning, the R1 input is weak but necessary if a D neuron with non-zero external input is to exceed the firing threshold and fire. Thus, any pattern activating a neuron in E layer - relevant or distractor - will fire its mirror neuron in the D layer.

The purpose of learning in Subtask I is to train D layer neurons to react only to relevant patterns in each contextual episode. The desired response to distractors is as follows: If they appear before the first relevant pattern in a contextual episode, no D neurons should fire; if they appear after one or more relevant patterns, the D neuron corresponding to the last relevant pattern should remain active until the next relevant pattern comes at the E layer.

The primary locus of learning at this stage is the R1-to-D projection. The C-to-R1 projection activates neurons in the active set of the latent attractor associated with a contextual episode. Once triggered, this latent attractor then projects a stable *bias mask* on the D neurons via the R1-to-D connections such that D neurons corresponding to stimuli relevant in the current context are disposed towards activity, and the rest are inhibited. If (and when) a relevant stimulus excites the D layer, the corresponding neuron is able to fire, but distractor stimuli are unable to overcome the bias mask projected by R1 and cannot activate their D neurons. In the latter case, the recurrent connections in the D layer and the projection from R1 keeps the previously activated relevant stimulus neuron active until it is deactivated by a relevant stimulus firing another D neuron.

The *bias mask* is produced by long-term potentiation (LTP) and long-term depression (LTD) in the R1-to-D connections. Initially, the weights of the R1-to-D connections are set to random, small values. Co-active R1 and D neurons have their connections strengthened, while connections from active R1 neurons to inactive D neurons are depressed. Since distractors are chosen randomly for each presentation of a contextual episode and the relevant patterns appear reliably each time, connections to D neurons corresponding to relevant stimuli undergo relatively more LTP while LTD is spread more evenly.

Over time, this results in the desired biasing pattern from the latent attractor in R1 to the D layer.

1) Slow and Fast LTP/LTD: The R1-to-D projection serves two purposes: 1) To create the *bias mask* so that only D neurons of relevant patterns fire; and 2) Together with the self-feedback input, to keep an active D neuron corresponding to a relevant pattern firing until the subsequent relevant input comes in. These two requirements lead to two distinct types of LTP/LTD operating on the R1-to-D weights.

The fast LTP rule used is:

$$w_{ij}^{R1D}(\tau) = \begin{cases} min(\ w_{ij}^{R1D}(\tau-1)\ (\ 1+\alpha_{ltp}(\tau)\ ),\ w_{max}^{R1D}\ ) \\ \quad if\ x_i^D(t)x_j^{R1}(\tau)=1, \zeta < p_{LTP} \\ w_{ij}^{R1D}(\tau-1), \quad otherwise \end{cases} \tag{3}$$

where ζ is a uniform random variable and p_{LTP} is the probability of undergoing LTP. Learning is biased further by a variable learning rate $\alpha_{ltp}(\tau-1)$ with the magnitude of the weight:

$$\begin{aligned} \alpha_{ltp}(w) &= \frac{1}{2}(\alpha_{max}+\alpha_{min}) \\ &- \frac{1}{2}(\alpha_{max}-\alpha_{min})\frac{e^{\lambda f(w)}-e^{-\lambda f(w)}}{e^{\lambda f(w)}+e^{-\lambda f(w)}} \\ f(w) &= \frac{1}{w_{max}-w_{min}}(\mu(w-w_{min})-\nu(w-w_{max})) \end{aligned} \tag{4}$$

The variable learning rate is a $tanh$ function of the weight magnitude: A small weight undergoes a smaller increment than a larger weight. Since all weights start out with similar small values, the variable learning rate magnifies the LTP for connections that are incremented repeatedly — and are likely targeted to relevant D neurons — while mitigating the effect of occasional increments, which are probably produced by distractors. The values of constants μ and ν control the skewness of the $tanh$ function with respect to weight limits.

The fast LTD rule is similar:

$$w_{ij}^{R1D}(\tau) = \begin{cases} max(w_{ij}^{R1D}(\tau-1)\ (\ 1-\alpha_{ltd}(\tau)\ ),\ w_{min}^{R1D}\ ) \\ \quad if\ x_i^D(t)=0, x_j^{R1}(\tau)=1, \zeta < p_{LTD} \\ w_{ij}^{R1D}(\tau-1), \quad otherwise \end{cases} \tag{5}$$

The variable learning rate for LTD is set as that for LTP.

The fast LTP/LTD modifies R1-to-D connections every fast time-step, τ, after R1 neurons fire. It strengthens the connection between the response in R1 due to the D input at time t and the active D neuron ($x_i^D(t)$) at time $t(\tau)$. Over time, as relevant patterns appear more often than distractors, active relevant D neurons will receive a stronger R1 input than other D neurons. The R1 input will have an *indirect* self-excitatory role for D neurons, similar to that of the *direct* self-feedback from D layer onto itself. The difference is that the R1 self-excitatory input is not equal for all D neurons as is the self-feedback, but is higher only for relevant D neurons. Fast LTP/LTD ensures that an active relevant neuron in the detector layer remains active until the next relevant pattern.

In addition to the fast LTP/LTD described above, the connections from R1 to D are also modified by what we term *slow LTP/LTD*. This is done as in Eqns. (3) and (5), but using $x_j^{R1}(\tau-1)$ instead of $x_j^{R1}(\tau)$ and t instead of τ everywhere else. Slow LTP/LTD happens every time a D neuron fires (each τ_f steps), between the active D neuron at time t and the activity in R1 layer at the previous τ time step ($\tau-1$). The purpose of the slow LTP/LTD is to create the *bias mask* of the R1 input onto relevant D neurons. Over time, most R1 neurons in the active set of the selected R1 attractor will be more strongly connected to relevant D neurons, and very weakly connected to distractor D neurons.

The slow LTP/LTD will slowly begin to block distractor neurons in the detector layer from firing. This is because relevant patterns are presented more often than distractors, and because of the variable LTP/LTD learning rate which keeps small weights down and increases larger weights. In the model, the LTP learning rate is bigger than that of LTD. Also LTD rate is kept constant.

At the end of Subtask I, the output of the detector layer in each contextual episode is a sequence of only relevant patterns. Each relevant pattern activates its detector layer neuron and stays on until the next relevant pattern comes in at the E layer. Distractor inputs before first relevant pattern in a contextual episode do not activate any detector layer neurons.

B. Subtask II: Learning the Order of Relevant Patterns

Once the D layer reliably eliminates distractors from the input sequence, the second subtask starts: Learning the order of the relevant patterns in each contextual episode. Subtask II has two phases: (a) learning and (b) retrieval. During (a), the system learns the order of relevant patterns. The input in this phase is similar to that of Subtask I: A set of contextual episodes presented repeatedly at the external (E) layer. In (b), the system is tested by checking if it can sequentially activate the relevant D neurons for each contextual episode in the right order. The input in this phase is just the context pattern in the C layer — signaling the beginning of a different context. There is no activity in the E layer.

For Subtask II, a latent attractor module (LAN2) is introduced (see Figure 1). LAN2 is similar to LAN1 having two layers: Response 2 (R2) and Hidden 2 (H2) connected through disynaptic connections. The same number of latent attractors is stored in LAN2 as in LAN1. C layer projects onto R2 the context pattern of each episode. The C-to-R2 connections are set like the C-to-R1 connections: The initial random weight values undergo LTP to associate the context pattern of each episode with the corresponding attractor.

The D and R2 layers are connected disynaptically. During phase (a), both connections (D-to-R2 and R2-to-D) undergo LTP/LTD. The R2-to-D projection is not active during this phase. In phase (b), the E layer is inactive, and firing in the D layer is determined by the R2 and R1 inputs. The R1-to-D projection provides the *bias* mask onto only the relevant D neurons in, while the R2-to-D projection has to select the next relevant D neuron in the sequence.

While the input to the E layer is the same as in Subtask I — relevant patterns mixed with distractors — the activity in D layer substitutes distractors with relevant patterns because of the prior Subtask I learning. Thus, the D input onto R2 is a sequence of relevant patterns, where each pattern is active one or more times in a row. A fast LTP/LTD process similar

to the one used in the R1-to-D connections (relations (3-5)) takes place in the D-to-R2 weight. The purpose is to ensure that each active relevant D neuron will reliably select a distinct pattern of activity in R2.

The R2 responses to an active relevant D neuron, in turn, activate the next relevant D neuron in the sequence. The LTP rule in the R2-to-D connections accomplishes this by associating the activity in R2 — due to the qth relevant pattern in the canonical sequence of a contextual episode (or to the context pattern input) – to the $q+1$th (or first) relevant D neuron, as shown in the relation below:

$$w_{ij}^{R2D}(t) = \begin{cases} min(w_{ij}^{R2D}(t-1)\ (\ 1+\alpha_{ltp}(t)\ ),\ w_{max}^{R2D}) \\ \quad if\ x_i^D(t)x_j^{R2}(\tau-1)=1, x<p_{LTP}, n_f=1 \\ max(w_{ij}^{R2D}(t-1)\ (\ 1-\alpha_{ltp}(t)\ ),\ w_{min}^{R2D}) \\ \quad if\ x_i^D(t)x_j^{R2}(\tau-1)=1, x<p_{LTP}, n_f>1 \\ w_{ij}^{R2D}(t-1),\quad otherwise \end{cases} \tag{6}$$

with n_f the number of times a D layer neuron fires repeatedly and $\alpha_{ltp}(t)$ the variable rate of LTP (relation (4).) Essentially, the rule updates a weight connecting two co-active D and R2 neurons as follows: When the D neuron fires a series of consecutive spikes, the weight goes up on the first spike and down on subsequent spikes. The magnitude of the change ($\alpha_{ltp}(t)$) depends on the value of the weight. Over time, the rule assures that the R2 response to the qth relevant pattern in an episode will project strongly only onto the D neuron corresponding to the $(q+1)$th relevant neuron in the sequence.

The sequence in a contextual episode is recalled in the retrieval phase (b) as follows: The R2 response to the context pattern in C layer selects the D neuron for the first relevant pattern. Then that one, in turn, activates a pattern in R2, firing the D neuron for the second relevant pattern, and so on until the whole sequence is recalled.

The dendritic input into i neuron in R2 is as given in relation (1) but with R2(H2) instead of R1(H1). The excitation of a D layer neuron, i, in the learning phase (a) is the same as in relation (2). The dendritic sum of neuron i in D layer in the retrieval phase (b) is:

$$\begin{aligned} h_i^D(t) &= g_{DD}\sum_{j\in D} w_{ij}^{DD}x_j^D(t-1) + g_{R1D}\sum_{j\in R1} w_{ij}^{R1D}x_j^{R1}(\tau) \\ &+ g_{R2D}\sum_{j\in R2} w_{ij}^{R2D}x_j^{R2}(\tau) \end{aligned} \tag{7}$$

IV. Simulation results

We simulated a system with these values: $N_E = N_D = 20$, $N_{R1} = N_{R2} = 1000$, $N_{H1} = N_{H2} = 500$, $N_C = 500$, $\tau_f = 3$. LAN1 and LAN2 had both 5 latent attractors with an active set of $A_{R1} = A_{R2} = 100$ and $A_{H1} = A_{H2} = 50$ and $K-of-N$ values: $K_{R1} = 60$, $K_{R2} = 40$, $K_{H1} = K_{H2} = 45$. Connection probabilities from R1-to-H1 and R2-to-H2 were 0.7, from H1-to-R1 and H2-to-R2, 0.9, from C-to-R2 and C-to-R1 0.8, from D-to-R1, 0.8, D-to-R2, 0.4, and from R1-to-D and R2-to-D, 0.8. Simulations were done with N_e=4 contextual episodes, each with a single context pattern each, N_r= 4 relevant symbols and $N_d = 4$ distractors. Relevant symbols were chosen randomly for each contextual episode at the beginning of simulation from a set of 20 symbols, while distractor symbols were chosen randomly from the same set each time an episode was presented at the input.

Figure 2 shows the performance on Subtask I measured by: $L_{q1}(T) = (N_c(T) - N_r)/(L_s - N_r)$, with $N_c(T)$ the number of correct firings in D layer at the Tth presentation of a contextual episode, L_s the length of an input sequence. It can be seen that $L_{q1}(T)$ increases slowly over time. At the end there are still a few situations when some firings are incorrect. Those are detailed next.

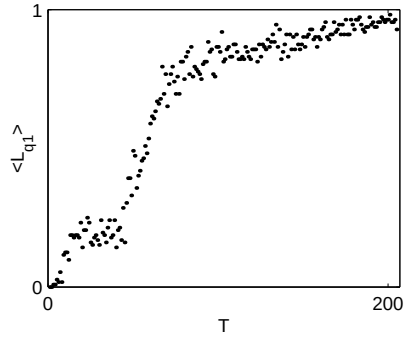

Fig. 2. Average learning quality ($< L_{q1} >$) of Subtask I over time. The mean is over seven simulations and four episodes in each simulation.

Figure 3 shows the learning dynamics during Subtask I: $I_d(T)$ is the mean percentage of distractor inputs that activate their D neuron, and $I_{d0}(T)$ is the mean percentage of distractor inputs — after the first relevant input in an episode — that do not activate any D layer neurons. Both situations represent incorrect firings. The plots show that the system learns to neglect distractors very rapidly (i.e., they cannot fire their own D neuron), but longer to stabilize the activity of the correct D neurons (i.e., preventing distractors from firing other D neurons).

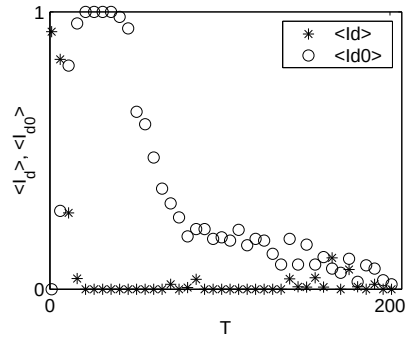

Fig. 3. Mean I_d and I_{d0} as a function of time. The mean is over seven simulations and four episodes in each simulation

Figure 4 shows the quality of retrieval phase (b) in Subtask II: $L_{q2}(q) = N_p(q)/N_e$, where $N_p(q)$ is the number of

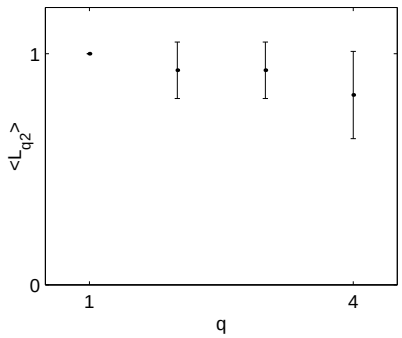

Fig. 4. Mean and standard deviation of L_{q2} values over seven simulations and four episodes as a function of q, the index of the relevant pattern.

correct firings of the qth relevant D neuron over all contextual episodes. The plot shows the mean and standard deviation $L_{q2}(q)$ over seven simulations and four episodes in each. The first pattern of each episode in each simulation is always recalled. Later patterns are recalled most of the time, but there are occasional failures. Those episodes are the ones that had incorrect firings of type $I_{d0}(T)$.

V. Conclusions and Discussion

We have shown how a neural system with simple Hebbian style LTP/LTD rules is able to discover sequences of relevant patterns embedded in distractors over multiple contextual episodes, and to learn the order between them. The system is given no knowledge of which patterns are relevant or distractors and what their order is. Distractors in one episode can be relevant patterns in others. The only information available to the system is the beginning of a distinct contextual episode. The central element in the model is the latent attractor network that keeps the representation of relevant and/or distractor patterns in different contextual episodes apart. Several issues in the model — e.g., the sequencing of subtasks — require further study, which will be addressed in the future.

References

[1] Y. Baram. Memorizing binary vector sequences by a sparsely encoded network. *IEEE Trans. on Neural Networks*, 5:974–981, 1994.

[2] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. on Neural Networks*, Vol. 5, No. 2:157–166, 1994.

[3] T.V.P. Bliss and T. Lømo. Long-lasting potentiation of synaptic transmission in the dentate area of the anesthetized rabbit following stimulation of the perforant path. *J. Physiol. (Lond.)*, 232:331–356, 1973.

[4] J.-C. Chappelier, M. Gori, and A. Grumbach. Time in connectionist models. In R. Sun and C.L. Giles, editors, *Sequence Learning: Paradigms, Algorithms, and Applications*, pages 105–134. Springer-Verlag, Berlin, 2000.

[5] S. Doboli and A.A Minai. Progressive attractor selection in latent attractor networks. In *Proc. of IJCNN'01, Washingto, D.C., USA*, 2001.

[6] S. Doboli and A.A Minai. Latent attractor selection in the presence of irrelevant stimuli. In *World Congress on Computational Intelligence, (WCCI'2002, Hawaii, USA*, 2002.

[7] S. Doboli and A.A Minai. Latent attractor selection for variable length episodic context stimuli with distractors. In *Proc. of IJCNN'03, Portland, OR, USA*, 2003.

[8] S. Doboli and A.A Minai. Network capacity analysis for latent attractor computation. *Network:Comput. Neural Syst.*, 14:273–302, 2003.

[9] S. Doboli, A.A. Minai, and P.J. Best. A latent attractors model of context selection in the dentate gyrus-hilus system. *Neurocomputing*, 26-27:671–676, 1999.

[10] S. Doboli, A.A. Minai, and P.J. Best. Latent attractors: a model for context-dependence place representations in the hippocampus. *Neural Computation*, 12(5):1009–1043, 2000.

[11] S.M. Dudek and M.F. Bear. Bidirectional long-term modification of synaptic effectiveness in the adult and immature hippocampus. *J. Neurosci.*, 13:2910–2918, 1993.

[12] J.L. Elman. Finding structure in time. *Cog. Sci.*, 14:179–211, 1990.

[13] P. Frasconi and M. Gori. Computational capabilities of local-feedback recurrent networks acting as finite-state machines. *IEEE Trans. on Neural Networks*, Vol. 7, No. 6:1521–1525, 1996.

[14] P. Frasconi, M. Gori, and G. Soda. Local feedback multilayered metworks. *Neural Computation*, 4:120–130, 1992.

[15] C.L. Giles, C.B. Miller, D. Chen, H.H. Chen, G.Z. Sun, and Y.C. Lee. Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4:393–405, 1992.

[16] R. Granger, J. Whitson, J. Larson, and G. Lynch. Non-hebbian properties of long-term potentiation enable high-capacity encoding of temporal sequences. *Proc. Nat. Acad. Sci. USA*, 91:10104–10108, 1994.

[17] I. Guyon, L. Personnaz, J.P. Nadal, and G. Dreyfus. Storage and retrieval of complex sequences in neural netwoks. *Phys. Rev. A*, 38:6365–6372, 1988.

[18] T.M. Heskes and S. Gielen. Retrieval of pattern sequences at variable speeds in neural network with delays. *Neural Networks*, 5:145–152, 1992.

[19] M.I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proc. 8th Conference of the Cognitive Science Society*, pages 531–546. Lawrence Elbaum, 1986.

[20] W.B. Levy and O. Steward. Synapses as associative memory elements in the hippocampal formation. *Brain Res.*, 175:233–245, 1979.

[21] W.B. Levy and X. Wu. The relationship of local context cues to sequence length memory capacity. *Network*, 7:371–384, 1996.

[22] A.A. Minai, G.L. Barrows, and W.B. Levy. Disambiguation of pattern sequences with recurrent networks. In *Proc. WCNN, San Diego*, volume IV, pages 176–180, 1994.

[23] A.A. Minai and P.J. Best. Encoding spatial context: A hypothesis on the function of the dentate gyrus-hilus system. In *Proc. Int. Joint Conf. on Neural Networks, Anchorage*, pages 587–592, 1998.

[24] M.C. Mozer. A focused backpropagation algorithm for temporal pattern recognition. *Complex Systems*, 3:349–381, 1989.

[25] R.M. Mulkey and R.C. Malenka. Mechanisms underlying induction of homosynaptic long-term depression in area CA1 of the hippocampus. *Neuron*, 9:967–975, 1992.

[26] D.E. Rumelhart and D. Zipser. Feature discovery by competitive learning. *Cognitive Sci.*, 9:75–112, 1985.

[27] H. Sompolinsky and I. Kanter. Temporal association in asymmetric neural networks. *Phys. Rev. Lett.*, 57:2861–2864, 1986.

[28] D.L. Wang and B. Yuwono. Incremental learning of complex temporal patterns. *IEEE Trans. on Neural Networks*, 7:1465–1481, 1996.

[29] L. Wang. Multi-associative neural networks and their applications to learning and retrieving complex spatio-temporal sequences. *IEEE Trans. on Systems, Man and Cybernetics - Part B: Cybernetics*, 29:73–82, 1999.

[30] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavio ral Sciences*. PhD thesis, Harvard University, 1974.

[31] R.J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

[32] C.H. Wu. Artificial neural networks for molecular sequence analysis. *Computers & Chemistry*, 21:237–256, 1997.

[33] M.J. Zaki. Sequence mining in categorical domains: Algorithms and applications. In R. Sun and C.L. Giles, editors, *Sequence Learning: Paradigms, Algorithms, and Applications*, pages 162–187. Springer-Verlag, Berlin, 2000.