Chapter 1

LATENT ATTRACTORS: A GENERAL PARADIGM FOR CONTEXT-DEPENDENT NEURAL COMPUTATION

Simona Doboli
Computer Science Department
Hofstra University, Hempstead, NY 11549, USA
Simona.Doboli@hofstra.edu

Ali A. Minai
Department of Electrical & Computer Engineering and Computer Science
University of Cincinnati, Cincinnati, OH 45221, USA
Ali.Minai@uc.edu

Abstract Context is an essential part of all cognitive function. However, neural network models have only considered this issue in limited ways, focusing primarily on the conditioning of a system's response by its recent history. This type of context, which we term Type I, is clearly relevant in many situations, but in other cases, the system's response for an extended period must be conditioned by stimuli encountered at a specific earlier time. For example, the decision to turn left or right at an intersection point in a navigation task depends on the goal set at the beginning of the task. We term this type of context, which sets the "frame of reference" for an entire episode, Type II context. The prefrontal cortex in mammals has been hypothesized to perform this function, but it has been difficult to incorporate this into neural network models.

In the present chapter, we describe an approach called latent attractors that allows self-organizing neural systems to simultaneously incorporate both Type I and Type II context dependency. We demonstrate this by applying the approach to a series of problems requiring one or both types of context. We also argue that the latent attractor approach is a general and flexible method for incorporating multi-scale temporal dependence into neural systems, and possibly other self-organized network models.

Keywords: Attractor networks, recurrent networks, context, sequence learning, modular networks, multi-scale dynamics.

1. Introduction

The most important and unique attribute of cognition is the ability to learn autonomously — to infer the necessary categories, contexts, and relationships from brief, unstructured interactions with the environment. One very important characteristic of information in the real world is its temporal variability at multiple scales, and the effect this has on the interpretation of and response to subsequent information. An intelligent system must be able to apprehend, represent, and use this multi-level information in responding to its environment. This, in turn, requires a certain degree of structural and informational "depth" in the system, giving it the ability to hold and process information simultaneously at many levels. For example, a system may have to simultaneously condition its response on: 1) The immediate stimulus (integrated over a few hundred milliseconds); 2) Recent experience (over the past few seconds); and 3) Motivational and cognitive context (typically stable over minutes or hours). The system must account for all the different types of time-varying influences simultaneously and in an integrated way, using mechanisms that are generic and adaptive, not fixed *a priori*.

The aspect of multi-scale temporal processing we are interested in is the ability of a neural system to react to contexts of different extents and scales. In this work, we present a general framework of multi-scale context dependencies and investigate possible neural mechanisms to implement some of the required features.

2. Multi-Scale Context Dependence

We define context as the relevant information in the past that affects the current response of the system. The response of a context-dependent system depends on current input, while being conditioned on context:

$$y(t) = f(x(t)|x^c(t)) \tag{1}$$

The context is denoted as $x^c(t)$ and is the set of past inputs that, together with the current input $x(t)$, uniquely determines the response of the system $y(t)$. Thus, it is functionally equivalent to the concept of state in dynamical systems.

There are essentially two types of contexts. One, which we call *immediate context* or *type I context*, is formed by the immediate past from $t-1$ to $t-p$, with p fixed for any given t, but variable over time and

possibly unknown:

$$x^{cI}(t) = \{x(t-1), x(t-2), \dots\ x(t-p)\} \tag{2}$$

This type of context appears often in sequential processing tasks, where the response depends on recent inputs, up to a certain point, p, in the past. Neural implementation of type I context has been successfully handled with recurrent neural networks. Such recurrent architectures include networks with output feedback [35, 81, 71, 22, 27, 53, 49, 41, 42], local feedback in individual neurons [52, 24, 23, 5], or slowly decaying correlations of activity [63]. One interesting approach is to use two feed-forward networks with one network modulating the weights or gating the outputs of the other to produce context-dependence [70].

While recurrent schemes allow the networks to develop internal representations of stimulus/state history, they all show recency effects, and perform poorly if the context information is too remote in time from t. Of course, long tapped delay lines can handle arbitrarily long dependencies, but only at the cost of additional delay elements and not with any flexibility.

Type I context occurs in many important situations including trajectory learning [35, 81, 58], speech or text recognition [22], learning automata [71, 27], and sequence learning [63, 49, 41, 42, 78–80].

Type II or episodic context, is defined as a finite set of past input at a *fixed* time in the past, such that the system's response, $y(t)$, depends on it regardless of t. Thus, it is defined by:

$$x^{cII}(t) = \{x(k_1), x(k_2), \dots\ x(k_r)\} \tag{3}$$

with $k_i,\ i = 1, \dots\ r$ denoting a set of past inputs unrelated to current time t, with r finite. In this case, the response of the system is conditioned on a set of inputs that happened sometime in the past, independent of the time interval between them and the current input. Examples of episodic context include the conditioning of responses by goals, social situations, location, time of day, etc. For example, one would answer a ringing phone in one's own office, but not in someone else's, or turn left or right at a particular intersection depending on the destination set at the beginning of a trip. Type II context corresponds roughly (but not exactly) to the notion of the *working memory* function thought to be mediated by the prefrontal cortex [48, 20, 68, 67, 57].

Type II context cannot be captured easily with recurrent neural networks [4]. Its coding requires long-term "latching", which is fundamentally different from transient or window-based short-term memory used for type I context. Indeed, a classic problem involving Type II context is the *latching problem* described by Bengio et al. [4].

An interesting approach to long-term context is the LSTM network proposed by [31]. The network is a recurrent network with hidden units called memory cells that through a mechanism of input and output gating prevent the gradient based error from decaying, and thus are able store dependencies to more temporally distal inputs than typical recurrent networks can. The learning algorithm is similar to real time recurrent learning (RTRL). LSTM has been shown to solve a variety of long-term sequence problems [31]. Another recent approach [68, 57, 67] maps a reinforcement learning algorithm on the architecture and functionality of the prefrontal cortex and basal ganglia. The model is able to learn simple Type II context problems much faster than general recurrent networks. Again, the method employs a dynamic and adaptive gating mechanism that learns the relevant context information and keeps it active in the presence of distractors.

A *multi-scale context-dependent system* is one that is conditioned on both type I and type II contexts:

$$y(t) = f(x(t)|x^{cI}(t), x^{c}II(t)) \tag{4}$$

This represents the most general case of sequential processing, where the response depends on current input, is conditioned on what happened in the near past ($x^{cI}(t)$), and also on the episodic context specified at some fixed earlier time ($x^{cII}(t)$). A supervised recurrent network designed for handling both types of dependencies is proposed in [4]. It uses hidden units operating at different time scales, thus allowing the gradient based error to decay rapidly for short-term context or slowly for long-term context.

Examples of multi-scale systems are speech generation (the word to be generated at time t depends on previous words in the same sentence, but also on the global discussion context set at the beginning of the conversation), spatial navigation (the next action depends on the previous actions, but also on the overall goal), etc.

Sometimes it may be appropriate to hypothesize that type I and II context are signaled by distinct input streams in the neural system, but in general, such a dichotomy cannot be assumed. For example, both types of context-dependence in hippocampal place fields seems to depend on the same afferent inputs [17]. The goal in our work is to investigate whether a neural system using biologically plausible processes (e.g., no supervised learning) can meet the following requirements: extract and encode multi-scale context from a single input stream. Requirements for a multi-scale context-dependent neural system (4) are:

1 Learn, encode and recognize both type I and type II contexts continuously from the *same* input stream (e.g., no separate context inputs) feeding sequentially into the system.

2 Handle continuous change in the order of type I context (p in 2).

3 Use type II context independent of when this context was perceived at the input.

In addition to these immediate concerns, the use of context-dependence in cognitive information processing also raises the issues of how context is represented and how it is related to the equally important issue of attention. In particular, it seems more useful to see context as a *dynamic process*, adaptively configuring the response-generating pathways of a cognitive system in real time, essentially playing the same role in response as attention plays in sensing.

Motivated by the functionality of the hippocampus, we have previously [50, 17] proposed a novel paradigm called *latent attractors* for type II context-dependence. In this model, it was assumed that each type II context characterizes an episode where the system generates responses to stimuli. The context is recognized at the beginning of the episode and must condition the responses of the system until switched by the recognition of a new context. Beginning with a simple initial model where context-recognition was triggered by a single initial stimulus pattern in each episode, we extended the system to the general case of type II systems (3), where more complex, noisy spatiotemporal patterns are used as triggers and the system is able to recognize and settle into the correct context progressively[9–11]. A more general model – also based on latent attractors – was proposed in [13]. This is a multi-scale context-dependent system as described in 4, though with a restrictive form of type I context. The system learned simple sequences in different type II contexts, embedded within arbitrary distractor patterns.

In this chapter, we start with a summary of our previous findings and generalize them into a single system that responds to type I context of order 1 and to type II context of the general form given in 3.

3. Latent Attractors

The primary challenge for any model of episodic context is the following: How can a system's response be made dependent simultaneously on episodic context and the current stimulus in a way that preserves the essential information in the latter? It is easy enough to simply produce arbitrarily different responses for every stimulus in each context, but that is not sufficient. Stimuli to a cognitive system represent the

structure of some information space whose state they encode (e.g., sensory cues in an environment). The similarity or dissimilarity between stimulus patterns, therefore, represents useful information that should be reflected in the response. Any context-dependence mechanism must simultaneously allow preservation of this information. What is needed, therefore, is not a "mixture" of context with stimulus information, but a way for context to *influence* the natural flow of information without disrupting it. This may be termed the *Principle of Contextual Influence*. Latent attractors are an embodiment of this principle.

Latent attractors (LA) networks are a novel neural network paradigm proposed as a mechanism for episodic context dependence in spatial tasks [50, 16, 15, 17]. The concept was inspired by the architecture of the rat hippocampus and by its functionality as revealed experimentally in spatial navigation tasks. It explained how different spatial representations (place codes [56]) can arise in two environments with identical sensory cues but representing distinct contexts (e.g. two distinct tasks in the same environment, two identical rooms with different entry points, etc.) [60, 45, 66].

From the network architecture viewpoint, LA networks belong to the class of *attractor networks*, used widely as models of associative memory [46, 32, 33]. These are networks with recurrent connectivity where specific patterns of activation are embedded as attractors for the network dynamics by appropriate choice of connection weights — typically through some variant of Hebbian learning. When the network is allowed to relax from an initial state near an attractor, it recovers the corresponding stored pattern of activity. Attractor networks have also been generalized to store sequences rather than fixed points[75, 40, 49, 41, 42, 44], and have been used to model a variety of cognitive functions, including those of the hippocampus[65, 40, 49, 41, 42, 6, 62, 61, 73, 69, 44, 74]. Attractor networks are among the most useful generic paradigms in neural systems as well as in the broader class of dynamic network (e.g., gene regulation networks [36, 72]).

The primary distinction between conventional attractor networks and latent attractor networks is in the role of the attractors themselves. In standard attractor networks, recovery of the attractor — fixed point or periodic — through recurrent dynamics is the goal of the network dynamics, and the pattern represented by the attractor is fully manifested under ideal conditions. In contrast, a latent attractor is not meant to be manifested explicitly, but to act as a *dynamically stabilized bias* on the response of the network to an incoming stimulus. When a latent attractor is triggered (see below), the effect is to recurrently bias a specific subset of neurons, termed the *active set (AS)* towards activity and

to inhibit the rest (Figure 1.1). However, it is the feed-forward stimulus that determines *which* neurons among the AS actually become active, thus creating a response dependent on the stimulus but conditioned by the currently active latent attractor, which represents the type II context. The active sets associated with different latent attractors are chosen randomly as a small fraction of the total neurons in the network, which means that they are not mutually exclusive. We have previously reported detailed analysis of LA networks in terms of capacity and performance[12].

Referring to the *Principle of Contextual Influence* defined earlier, a useful way to think of latent attractors is as a generic, adaptive mechanism for embedding and selecting sub-networks or processing modules within a larger network in a controlled way — essentially "soft-assembling" a response-generation system on the fly based on context information. This viewpoint emphasizes the vast possibilities offered by such networks as building blocks of much more complex modular systems with combinatorial context-switched dynamics.

4. Implementation of Latent Attractor Networks

Latent attractors can embedded in one- or two-layer recurrent networks, and we have previously investigated and compared the properties of both [14], showing that two-layer LAs are better at simultaneously maintaining both type II context-dependence as well as dependence on current input. Here, we describe the two-layer implementation with the network architecture shown in Figure 1.1.

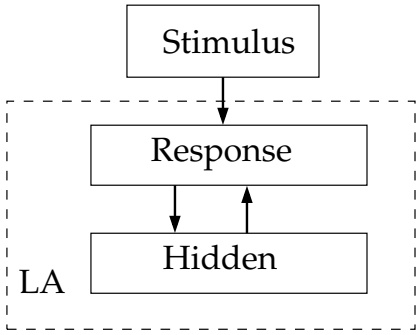

Figure 1.1. A two-layer LA network.

The system has a stimulus layer (S), a response layer (R) and a hidden layer (H) with N_S, N_R and N_H neurons, respectively. The neurons in

the input layer receive external input and project with connection probability p_{RI} to R layer neurons. The strength of the connections is also set randomly within an interval. The R layer projects to the H layer through randomly chosen connections with connection probability p_{HR}, and the H layer projects back to the R layer randomly, with connection probability p_{RH}. The strengths of the connections between R and H layers are set such that a set of M attractors is stored on the H-R sub-network. The M attractors are composed of two binary patterns, one in layer R and the other in layer H. The layer R pattern, $R^\alpha = \{r_1^\alpha r_2^\alpha ... r_{N_R}^\alpha\}$, has G_R active neurons forming the active set of α — denoted A_R^α — and $N_R - G_R$ inactive ones. Similarly, the layer H pattern, $H^\alpha = \{h_1^\alpha h_2^\alpha ... h_{N_H}^\alpha\}$, has G_H active neurons (set A_H^α). The attractors are embedded in the connections by setting the weights according to a clipped binary Hebbian rule first proposed by Willshaw [82]. The weight, w_{ij}^{RH}, from neuron $j \in H$ to $i \in R$ is set as $w_{ij}^{RH} = c_{ij}\Theta\left(\sum_\alpha r_i^\alpha h_j^\alpha\right)$, where c_{ij} is a binary variable indicating the existence of the j-to-i connection, and $\Theta(x) = 1$ if $x > 0$ and 0 else. Similarly, the weights from $j \in R$ neurons to $i \in H$ neurons are set to ($w_{ij}^{HR} = w_{ji}^{RH}$), producing a symmetric weight matrix overlaid by a random, sparse connectivity. When the network relaxes from an initial condition, the tendency is for the activity to get trapped in the active sets, A_R^α and A_H^α, of a particular attractor, α.

The network dynamics is synchronous and competitive. The weighted *dendritic inputs* for neurons $i \in R$ and $j \in H$ at time t are:

$$\begin{aligned} h_i^R(t) &= g_i(t)\sum_{j=1}^{N_H} w_{ij}^{HR} x_j^H(t-1) + \sum_{k=1}^{N_S} w_{ik}^{SR} x_k^S(t), \qquad (5) \\ h_j^H(t) &= \sum_{i=1}^{N_R} w_{ji}^{RH} x_i^R(t) \end{aligned}$$

where g_i is the recurrent gain of the $i \in R$ neurons and $x_{(.)}^{(.)}$ is the binary output of the neurons. The gain $g_i(t)$ parameterizes the relative strength of the recurrent and external inputs. The competitive firing rule allows only the K_R and K_H cells with the highest dendritic sums to fire in R and H, respectively.

In previous work [12], we have done extensive analytical and numerical investigation of the capacity of LA networks — defined as the capability of the system activity to remain confined in the active set of an attractor when faced with an indefinite number of stimuli that uniformly excite neurons in all attractors. It was found that, in general, the capacity of latent attractor networks is smaller than that of regular auto-associative

networks. This is because the latent attractor network allows only partial recall of the stored patterns, and the external input has to play a decisive role in choosing the activity pattern. In fact, from a standard attractor network viewpoint, the external input plays the role of noise, reducing stability, but that is part of the system's desired functionality.

5. Multi-Scale Context Dependence with Latent Attractors

In this section we describe some of the problems that have been addressed using latent attractor networks.

5.1 Single Pattern Type II Context Triggering

The original problem for which LAs were proposed can be stated as a general type II context dependence:

$$y(t) = f(x(t)|x^{cII}(t)) \tag{6}$$

where $x^{cII}(t)$ is defined in equation 3.

In terms of the LA network shown in Figure 1.1, $y(t)$ is the activation vector of size N_R over neurons in R layer: $y(t) = \{y_1(t), y_2(t), \ldots, y_{N_R}(t)\}$ and $x(t)$ represents the activity in the S layer: $x(t) = \{x_1(t), x_2(t), \ldots, x_{N_S}(t)\}$. The activity of each neuron is binary, 0 for inactive neurons, 1 for active neurons. In this model [50, 17, 12], $x^{cII}(t)$ is represented by a single *triggering pattern* of size N_S for each context. The active neurons in a triggering pattern are slightly more strongly connected to the R layer active set of one LA. When the triggering pattern for an attractor α is presented at the S layer, the initial activity in the R layer tends to be slightly higher in that attractor's active set, A_R^α, which preferentially fires neurons in the active set, A_H^α, in H. The strong recurrent connections between R and H thus project an almost uniform positive bias onto the A_R^α and A_H^α sets of neurons. Now a set of *regular input patterns* is presented to the network as input. These are patterns not pre-associated with any active set in the R layer and, on their own, they excite all M active sets equally. However, because the triggering pattern has biased the active sets of LA α, the competitive firing rule ensures that the $K_R < G_R$ neurons actually fired in R come preferentially from the G_R neurons of A_R^α, and the particular choice of fired neurons *within the active set* is determined by the external input, $x(t)$. The R layer activity fires $K_H \approx G_H$ neurons in the H layer, ensuring that it continues to project an almost uniform bias onto the active set A_R^α, ensuring the dependence of all R layer responses on the initial triggering pattern without loss of information from the current stimulus. This persists

until another triggering pattern appears as stimulus and switches the dynamics to another LA.

5.2 Multi-Pattern Type II Context Triggering

An extension to the problem in Section 5.1 [9–11] is a type II context defined by a set of context patterns, rather than just one triggering stimulus.

The network is presented with P different external stimulus sequences. Each sequence, S^q, of total length n

$$S^q = c_1^q[X^q]c_2^q[X^q]...[X^q]c_l^q[R^q] \quad q = 1, 2, ..., P \tag{7}$$

starts with a sub-sequence of r patterns – *the context sequence* – comprising l context patterns, c_i^q, interspersed randomly with $r - l$ non-context patterns, indicated by $[X^q]$. The last sub-sequence, $[R^q]$, comprises $n-r$ non-context stimuli and is called the *regular sequence*. The context patterns, c_i^q are drawn from a set of patterns called the *context set*, $C = \{c_k\}$, and the remaining patterns are drawn from the set, $R = \{r_k\}$. Both c_k, $r_k \in I$, where I is the input space of dimension N_S. For purposes of simulation, patterns in both sets are generated randomly, using the same procedure.

A total of n possible contexts are defined, each specified by a unique set of p context patterns drawn randomly without repetition from C. The context sequence for each sequence, S^q, includes context patterns for a unique context, albeit in random order and mixed with non-context patterns. Both the identity and position of non-context patterns in the context sequence is chosen randomly each time a context sequence is presented at the system input. This simulates gradual specification of the context based on specific cues sampled in random order and interspersed with non-cue inputs, e.g., recognizing a place by successively observing a set of landmarks in random order. The regular sequence for each S^q is also randomly chosen from R.

At the beginning of an episode, as context patterns are presented, each context pattern specifies the correct attractor with increasing certainty until, at the end of the context sequence, the attractor is uniquely identified. Thus, the network activity should gradually become confined to the active set of the correct context and remain confined during the presentation of the regular stimulus.

5.3 First-Order Multi-Scale Context-Dependence

As described earlier, multi-scale context-dependence requires that the system's response be conditioned by both type II and type I contexts. The input consists of P sequences of patterns or *episodes*. The temporal order of patterns in an episode E^q is given by:

$$E^q = C^q p_1^q p_2^q ... p_n^q, \;\; q = 1, ...P \tag{8}$$

where C^q is a context identifier chosen from a set C called the *context set* and p_i^q is a pattern chosen from another set V, of size N_V. Each episode has embedded in it a finite ordered set of patterns of length l – initially unknown to the system – called the *canonical sequence*: $S^q = s_1^q, s_2^q, \ldots s_l^q$, where $s_i^q \in V$. Canonical sequences are constructed by choosing patterns from V without repetition. The same pattern can be part of more than one canonical sequence. Once chosen, the canonical sequence of an episode remains fixed. The remaining $n - l$ patterns – called *distractors*– are chosen randomly from the set $V - S^q$, each time the episode E^q is presented at the input. Distractors represent the variable part of an episode with both their identity and their position chosen randomly each time the episode appears at the input. Over many experiences of an episode, the frequency of presentation of distractors is much lower than that of canonical patterns, which are part of each instance, always in the same order.

Pattern p_i^q in an episode is first given a randomly chosen role: distractor or canonical. Then its identity is determined, either from the canonical sequence S^q – in case p_i^q is a canonical pattern, or randomly chosen from the set $V - S^q$ – in case p_i^q is a distractor. The only constraint imposed on the placement of canonical patterns and distractors per episode is that no two consecutive canonical patterns are also consecutive in E^q. This eliminates the case where the order between canonical patterns can be detected statistically, without learning the role of each pattern – distractor or canonical. The task for the system is to learn the canonical sequence in each episode from sequences of canonical patterns embedded and intermixed with distractors, where neither the canonical patterns nor the distractors are identified explicitly. At the end of learning, the system must generate the canonical sequence when given the context identifier.

This problem is a type II problem, since each episode takes place in a different context. It is also a first-order type I problem, since the identity of the next canonical pattern depends only on that of the previous one.

Formally the problem can be stated as follows:

$$y(t) = f(x(t)|x(k), x(t-1)), \quad k < t \tag{9}$$

where k denotes the arbitrarily distant time in the past when the context stimulus was presented. Interestingly, this can also be seen as an on-line sequence mining task [84].

6. Network Models

In this section we detail two network models – both embedding the LA architecture shown in Figure 1.1. The first one is for the multi-pattern type II context-dependence presented in Section 5.2, and the second one for the first-order multi-scale dependence discussed in Section 5.3.

6.1 Network Model for Multi-Pattern Type II Context Triggering

The architecture of the network is shown in Figure 1.2. In addition to the system shown in Figure 1.1, there is an extra bias layer (B), with N_B neurons that receive random projections from S layer neurons with a p_{BS} probability of connection and project to R neurons with probability p_{RB}. The latent attractor module has the same functionality as described in Section 3.

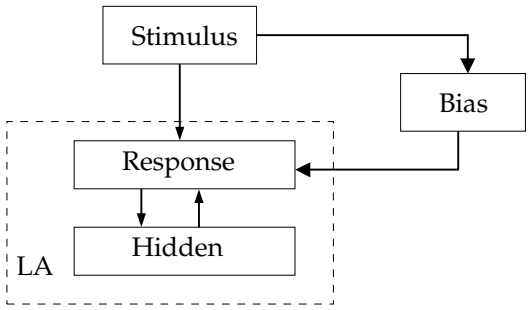

Figure 1.2. Network Model for Multi-Pattern Type II Context Triggering

Every context, q, is mapped to a latent attractor α. The S to R connections are set through associative learning such that active neurons in each sequence q context pattern C_i^q preferentially stimulate neurons in the active set of latent attractor α in the R layer. The associative rule adds a small amount, δ, to the random connection strength, w_{ij}^{RS},

from neuron $j \in S$ to $i \in R$ as follows:

$$w_{ij}^{RS} = \begin{cases} w_{ij}^{RS} + \delta & : \quad j \in C^q \ \ and \ \ i \in A_R^\alpha \\ w_{ij}^{RS} & : \quad otherwise \end{cases} \tag{10}$$

Since context patterns can overlap across different contexts, each context pattern, C_i^q, can become associated with several latent attractors. When the first context pattern, C_i^q, for a context q is presented at S, it biases the active sets of all latent attractors associated with it. Activity persists in these neurons (see below) until the next context input, C_2^q is presented. A combination of feed-forward and recurrent biasing ensures that activity now becomes confined to active sets of latent attractors associated with *both* C_1^q and C_2^q. In this way, as each context pattern is presented, it further confines activity to active sets of latent attractors associated with *all* context patterns presented so far until only the unique correct latent attractor is left active. This essentially implements a *sequential conjunctive mechanism* for context identification.

The role of the bias layer, B is to sustain the level of activity in the active sets of the candidate latent attractors during the presentation of distractor patterns in the context sequence. Each neuron in the B layer is associated with one context pattern. When a context pattern is presented at the stimulus layer S, the associated neuron in the biasing layer becomes active. The active biasing neuron, in turn, increases the excitation of the R neurons in the latent attractors that are associated with the corresponding context pattern. Thus, in between context patterns, the activity in the latent attractors tends to be preserved until a new context pattern is presented at the input. The activity of the biasing neurons is reset after a latent attractor has been fully activated (i.e. at the end of a context sequence). The biasing layer acts essentially as a working memory by sustaining the effect of context patterns until the context sequence is complete. Due to the new bias layer, the excitation of a R neuron i changes from equation (5) to:

$$h_i^R(t) = g_i(t) \sum_{j=1}^{N_H} w_{ij}^{HR} x_j^H(t-1) + \sum_{k=1}^{N_S} w_{ik}^{SR} x_k^S(t) + g_{bias} \sum_{l=1}^{N_B} w_{il}^{BR} x_l(t-1), \tag{11}$$

An important parameter in this network is the recurrent gain, $g_i(t)$, since it controls the stability of attractors by determining the strength of the recurrent projection to R relative to that of the external stimulus. For a latent attractor to remain active, neurons in its active set must have a $g_i(t)$ above a stability threshold value [17].

During the presentation of the context sequence, the network's state should not settle until it has received enough context patterns to uniquely

identify the sequence, and should not be disrupted by the intervening irrelevant stimuli. This is achieved through a process we term *incremental competitive positive feedback (ICPF)* [9, 10]. The stability of any attractor in the network is controlled by the recurrent gains, $g_i(t)$. When $g_i(t)$ values are small relative to the strength of the external projections, attractor dynamics is dominated by the impact of the feed-forward stimulus. If the stimulus is selectively associated with particular sets of neurons, these are likelier to win the competition for firing among R neurons. If the $g_i(t)$ values are large, the recurrent path dominates and the network is forced to choose between attractors due to competitive firing in R and H.

At the beginning of each episode, all $g_i(t)$ values are set below the stability threshold. As discussed above, the presentation of each context pattern must confine activity to the active sets of latent attractors associated with all the context patterns presented up to that point. This set of latent attractors is called the *candidate set*. The required behavior is produced by slowly increasing the recurrent gain $g_i(t)$ for neurons in the active sets of candidate set LAs as follows:

$$\hat{g}_i(t) = g_{min} + \frac{g_{max} - g_{min}}{(1 + e^{-\eta(a_k(t)-\beta)})}$$

$$d_i(t) = \hat{g}_i(t) - g_i(t-1)$$

$$g_i(t) = \begin{cases} \hat{g}_i(t) & if\ |d_i(t)| < \Delta g_{max} \\ g_i(t-1) + \Delta g_{max} sgn(d_i(t)) & else \end{cases} \tag{12}$$

where η is a rate of change parameter, β is an offset parameter, k is the index of the attractor for which i is in the active set, $a_k(t)$ is the total number of active neurons in the R active set of attractor k at time t, and g_{min} and g_{max} are, respectively, the minimum and maximum possible values of the recurrent gain. Thus, the gain seeks the nominal value $\hat{g}_i(t)$, but its absolute change is bounded by Δg_{max} ($\Delta g_{max} > 0$). The recurrent gain is set equal for all neurons in the R active set of LA k, proportional to the activity in that attractor at time t. Some neurons are in the active set of more than one LA. In this case, the gain of those neurons is set as the maximum of all gains from LAs in which it is part of the active set.

The modulation of recurrent gain on individual neurons is motivated by several biological considerations:

1 Projections to most cortical neurons are segregated on the dendritic tree, making the selective modulation of gain on input from individual sources quite possible [30].

2 It is well known that animals are especially attentive at the beginning of an episode, as indicated by the change in the hippocampal EEG theta rhythm. This leads to greater spike synchronization, lower firing latency, and other phenomena [51].

3 In the granule cells of the dentate gyrus, which corresponds roughly to our layer R, there is evidence [29, 34] of an intricate and highly specific system of excitability modulation based on motivation and attention [51].

4 The adaptation of the recurrent gain to all neurons in the active set of a latent attractor based on the global activity is supported by experimental results showing that the strength of distal synapses can be increased by the amount of local external excitation, independent of whether the postsynaptic neuron fired [47]. Moreover, it can be assumed that each neuron in the active set of an R attractor receives approximately the same excitation from the H layer, and this is, in turn, proportional to the activity in the active set of the R layer attractor.

6.2 Network Model for First-Order Multi-Scale Context Dependence

The network architecture for the problem in Section 5.3 is shown in Figure 1.3. Apart from the LA module there is an external layer (E), a detector layer (D), a context layer (C) and a predictor layer (P). All

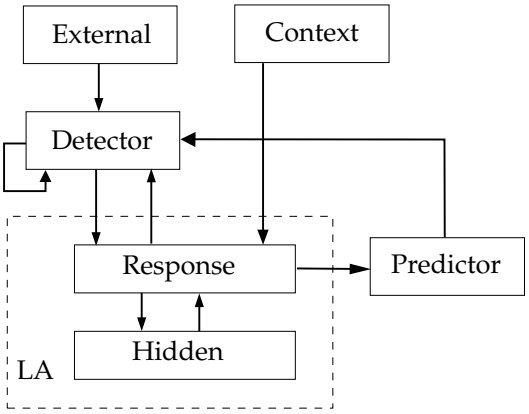

Figure 1.3. Network Model for First-Order Multi-Scale Dependence

projections – unless stated otherwise – are randomly chosen with a $p_{(.)}$ probability of connection.

The system functions in two modes: *Learning* and *recall*. In the learning mode, the input to the system is the episode described in equation 8. The first pattern of an episode – the context identifier – is activated at the C layer, which in turn selects the latent attractor for that episode in the R layer. The rest of the episode – the p^q patterns – are presented at the E layer. In the learning mode, the P-to-D connections are not active, but they undergo synaptic modifications [30]. In the recall mode, the only input received by the system is the context input coming from the C layer. This activates the correct attractor in the R layer, which, through R-to-P and, the now active P-to-D connections, starts the recall process. During recall, the patterns in the canonical sequence S^q are recalled in the correct order in the D layer.

The C layer has N_C neurons, and its inputs – the context identifiers – have K_C active neurons. The input from the C layer is active only at the beginning of an episode, and its role is to signal to the LA module a change in type II context. We assume that the context identifiers are already associated with a latent attractor through a process similar to the one described in Section 6.1: Small LTP adjustments to connection strengths between each context pattern and the active set of a latent attractor. The effect of the association is to switch the activity in the LA module each time a new context identifier appears at the C layer. Once the correct attractor is activated, the C input is set to zero, representing the situation where context is identified fully at the beginning of an episode or cognitive experience – e.g. spatial context – and remains in effect until the end of the episode.

The role of the E layer is to recognize all the p^q patterns in the episode, independent of whether they are canonical patterns or distractors. To keep the model simple, we use a 1-of-N code for the inputs, so that each pattern activates a single, unique neuron in E. The E layer projects to the D layer through one-to-one connections. The dimension of the E layer (N_E) equals the number of symbols in V set (see Section 5.3).

Coding in the D layer is also 1-of-N with one neuron active per input pattern and $N_D = N_E$. The role of the D layer is to filter out distractors from the incoming stimuli. At the beginning of learning, all patterns that fire neurons in E, fire their corresponding neurons in D as well. As learning advances, the D layer responds only to canonical patterns. Distractors are ignored, i.e., do not cause change of activity in D.

The D layer has two recurrent loops: The monosynaptic recurrent projection is a self-feedback of each D neuron onto itself, while the disynaptic feedback comprises D-to-R and R-to-D connections. The R-to-D

projection undergoes LTP/LTD during learning. The role of this loop is to bias only canonical patterns for activity in the D layer, and to inhibit distractors.

In the learning mode, the input to a D neuron is:

$$\begin{aligned} h_i^D(t) &= g_{ED} \sum_{j \in E} w_{ij}^{ED} x_j^E(t) + g_{DD} \sum_{j \in D} w_{ij}^{DD} x_j^D(t-1) \\ &+ g_{RD} \sum_{j \in R} w_{ij}^{RD} x_j^R(t-1) \end{aligned} \tag{13}$$

with $g_{(.)}$ denoting the gain of each connection. Activity in the D layer is a combination of winner-take-all and threshold firing: the neuron with the largest excitation above a threshold θ fires at each time step.

In recall mode, there is no E input and, functionally, the dominant projection is the one from the P layer. The input to a D neuron is:

$$\begin{aligned} h_i^D(t) &= g_{DD} \sum_{j \in D} w_{ij}^{DD} x_j^D(t-1) + g_{RD} \sum_{j \in R} w_{ij}^{RD} x_j^R(t-1) \\ &+ g_{PD} \sum_{j \in P} w_{ij}^{PD} x_j^P(t-1) \end{aligned} \tag{14}$$

The LA module behaves identically as shown in Section 3. Its main function is to generate context-dependent coding of the external patterns. Thus, the same pattern appearing in more than one episode produces distinct patterns of activity in the R layer. This allows multiple associations between the same canonical pattern and other canonical patterns to be learned by the P-to-D projection if they occur in different episodes.

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 R and H 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: $t(\tau) = \lfloor \tau/\tau_f \rfloor + 1$. 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.

The net input to neuron i in layer R is:

$$\begin{aligned} h_i^R(\tau) = g_{DR} \sum_{i \in D} w_{ij}^{DR} x_j^D(t) + g_{HR} \sum_{i \in H} w_{ij}^{HR} x_j^H(\tau - 1) + \\ + g_{CR} \sum_{i \in C} w_{ij}^{CR} x_j^C(t) \end{aligned} \tag{15}$$

where $g_{(.)}$ are the gains of each projection. The C input is active only at the beginning of an episode: $g_{CR} = 0$ for all $t > 1$ and the D layer input into R is active only for $t > 1$: $g_{DR} = 0,\ for\ t = 1$. Activity in the R and H layers is determined as described in Section 3. For notational simplicity we assume that $t = 1$ denotes the beginning of an episode.

P layer receives context-dependent coding of the external patterns via a very sparse $R-to-D$ projection. Its role is to predict – in recall mode – the next canonical pattern in the sequence. In the learning mode, the P-to-D projection is inactive, but undergoes LTP/LTD. Firing in the P layer is K winner take all, with the K_P most excited neurons firing.

6.2.1 Learning in the Model. Synaptic modifications take place during learning mode in $R - to - D$ and $P - to - D$ connections. Associations of context identifiers with the active set of a latent attractor in R are done before learning mode begins.

6.2.2 (a) Learning in *R*-to-*D* Connections. Initially, firing in the D layer is determined solely by the dominant E layer projection. R input provides uniform bias from the active set of the selected attractor onto all D neurons. Before learning, the R 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 layer E – canonical or distractor – will fire its mirror neuron in the D layer.

The purpose of learning in R-to-D projections is to train the D layer neurons to react only to canonical patterns in each contextual episode. Ideally, the latent attractor – once activated by the context identifier – projects a stable *bias mask* on the D neurons via the R-to-D connections such that D neurons corresponding to canonical stimuli in the current context are disposed towards activity, and the rest are inhibited. If (and when) a canonical pattern excites the D layer, the corresponding neuron is able to fire, but distractors are unable to overcome the bias mask projected by R and cannot activate their D neurons. In the latter case, the recurrent connections in the D layer and the projection from R keep the previously activated canonical stimulus neuron active until it is deactivated by the next canonical pattern in the sequence firing another D neuron. The *bias mask* is produced by long-term potentiation (LTP) and long-term depression (LTD) in the R-to-D connections.

There are two types of LTP/LTD in the network: *slow* and *fast* LTP/LTD. Slow LTP/LTD changes connection strengths each t_i step - i.e. associates $R(t_i - 1)$ activity with that of $D(t_i)$. Fast LTP/LTD acts every τ step and associates $R(\tau)$ with $D(t_i)$, where τ such that $t(\tau) = t_i$.

The fast LTP rule used is:

$$w_{ij}^{RD}(\tau) = \begin{cases} min(\ w_{ij}^{RD}(\tau-1)\ (\ 1+\alpha_{ltp}(\tau)\ ),\ w_{max}^{RD}\ ) \\ \quad if\ x_i^D(t)x_j^R(\tau)=1, \zeta < p_{LTP} \\ w_{ij}^{RD}(\tau-1), \quad otherwise \end{cases} \tag{16}$$

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

$$\begin{aligned} \alpha_{ltp}(w) &= \frac{1}{2}(\alpha_{max}+\alpha_{min}) \\ &- \frac{1}{2}(\alpha_{max}-\alpha_{min})\frac{e^{\lambda f(w)}-e^{-\lambda f(w)}}{e^{\lambda f(w)}+e^{-\lambda f(w)}} \\ f(w) &= \frac{1}{w_{max}-w_{min}}(\mu(w-w_{min})-\nu(w-w_{max})) \end{aligned} \tag{17}$$

The variable learning rate is a *tanh* function of the weight magnitude: If $\alpha_{max} > \alpha_{min}$, a small weight undergoes a smaller increment than a larger weight, otherwise a small weight undergoes a larger increment than a large 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 canonical 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:

$$w_{ij}^{RD}(\tau) = \begin{cases} max(w_{ij}^{RD}(\tau-1)\ (\ 1-\alpha_{ltd}(\tau)\ ),\ w_{min}^{RD}\ ) \\ \quad if\ x_i^D(t)=0, x_j^R(\tau)=1, \zeta < p_{LTD} \\ w_{ij}^{RD}(\tau-1), \quad otherwise \end{cases} \tag{18}$$

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

The fast LTP/LTD modifies $R-to-D$ connections at each fast (τ) time-step after the R neurons fire. It strengthens the connection between the response in R due to 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 in a particular context, active relevant D neurons receive a stronger R input than other D neurons. The R input 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 R self-excitatory input is not equal for all D neurons as is the self-feedback, but is higher only for relevant D neurons in a sequence. Fast LTP/LTD ensures that an active relevant neuron in the detector layer remains active until the next relevant pattern.

6.2.3 Slow LTP/LTD. In addition to the fast LTP/LTD described above, the connections from $R-to-D$ are also modified by what

we term *slow LTP/LTD*. This is done as in Equations (16) and (18), but using $x_j^R(t-1)$ instead of $x_j^R(\tau)$ 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 active neurons in the R layer at the previous t time step $(t-1)$. Slow LTP/LTD creates the *bias mask* of the R input onto canonical D neurons. Over time, most R neurons in the active set of the selected R attractor will be more strongly connected to canonical D neurons, and very weakly connected to distractor D neurons.

6.2.4 Analysis of Learning Dynamics in *R*-to-*D* Connections. The probability of a canonical pattern for Episode E^q appearing in the episode is 1, while the probability of a distractor d at any of $(n-l)$ positions in E^q is: $P(d) = 1/(N_V - l)$ with expectation $E(d) = (n-l)P(d)$ much lower than 1. Fast LTP/LTD uses this difference, to increase the R-to-D feedback input onto canonical D neurons compared to distractor D neurons.

Slow LTD has, initially, a decreasing effect for R-to-D connections onto both distractor and canonical D neurons, but larger for distractors. Slow LTP does not have any effect initially, since there are no consecutive canonical patterns in E^q (see Section 5.3). Slow LTP is effective only after canonical D neurons are able to sustain firing until the next canonical pattern is activated at E layer. This happens after fast LTP/LTD has increased the R input to the active D neuron at $t-1$ enough so that, in conjunction with self-feedback, it is able to win the firing competition in D. Only then is the association between R activity due to the canonical D neuron for s_i^q and the active canonical D neuron for s_{i+1}^q increased systematically. Before that, slow LTP strengthens random associations between the R response at $t-1$ to a p_i^q pattern and active D neuron to $p_{(i+1)}^q$. After distractor D neurons are not able to win the competition for firing, their R input undergoes even more slow LTD than before, increasing the *bias mask* effect.

The combined effect of fast LTP/LTD and slow LTP/LTD causes the following effects over time: 1) Due to fast LTP/LTD, canonical D neurons active at $t-1$ are able to remain active when a distractor is presented at E layer; 2) The *bias mask* slowly establishes as a consequence of slow LTP/LTD, thus effectively allowing only canonical patterns to activate their D neuron.

At the end of the learning mode, the D layer output in each contextual episode is a sequence of only relevant patterns. Each canonical pattern activates its detector layer neuron, which stays on until the next relevant

pattern comes in at the E layer. Distractor inputs are unable to change activity in layer D.

6.2.5 (b) Learning in *P*-to-*D* Connections. Learning in the P-to-D connections takes place concurrently with that in the $R-to-D$ connections, but does not affect the net input of a D neuron in this mode. The role of these connections is to associate the context-dependent activity in P caused by canonical input s_i^q to the canonical D neuron for $s_{(i+1)}^q$. They are active only in the recall mode.

Both, LTP and LTD occurs in the P-to-D connections. The LTP rule is:

$$w_{ij}^{PD}(t) = \begin{cases} min(w_{ij}^{PD}(t-1)\ (\ 1+\alpha_{ltp}(t)\ ),\ w_{max}^{PD}) \\ \quad if\ x_i^D(t)x_j^P(t-1) = 1, \zeta < p_{LTP}\ and\ n_f = 1 \\ max(w_{ij}^{PD}(t-1)\ (\ 1-\alpha_{ltp}(t)\ ),\ w_{min}^{PD}) \\ \quad if\ x_i^D(t)x_j^P(t-1) = 1, \zeta < p_{LTP}\ and\ n_f > 1 \\ w_{ij}^{PD}(t-1),\ \ otherwise \end{cases} \tag{19}$$

with n_f the number of times a D neuron fires repeatedly and $\alpha_{ltp}(t)$ the variable rate of LTP (Equation 17). Essentially, the rule updates a weight connecting two co-active D and R 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 R response to the ith canonical pattern in an episode will project strongly only onto the D neuron corresponding to the neuron for the $(i+1)$th canonical pattern in the sequence. The LTD rule is similar to that of slow LTD.

7. Simulation Results for Multi-Pattern Type II Context Triggering

Simulation were done using the model in Figure 1.2 with the following parameters: $N_S = 400$, $K_S = 40$, $p_S = 0.4$, $N_B = 20$, $p_{BS} = 0.9$, $p_{RB} = 0.9$, $N_R = 2000$, $G_R = 200$, $K_R = 40$, $N_H = 500$, $G_H = 50$, $K_H = 45$. There are $M = 10$ attractors embedded in the connections between the R and H layers. The modulation rate for recurrent gain g_i is $\eta = 0.5$ and $\beta = 33$. The gain of the B-to-R projection is $g_{bias} = 6$.

The context set C has 20 distinct patterns, from which $P = 5$ context sequences, C^q, are selected, each consisting of $l = 5$ distinct patterns picked randomly without repetition from C. Context patterns in different C^q are *not* mutually exclusive. Each C^q set is associated with a randomly chosen attractor by potentiating the connections from S to R layers according to equation 10. Also, each context pattern, C_k is associated with a neuron, k, in layer B through Hebbian potentiation

of the connections from the stimulus layer S and layer B. In turn, each biasing neuron, k, provides excitation to neurons in the active sets of those latent attractors whose context sequences C^qs include pattern C_k.

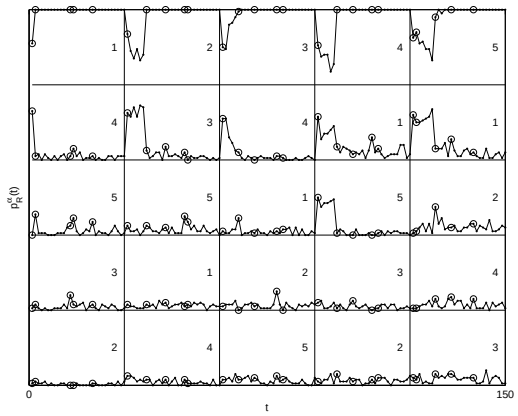

Figure 1.4. Normalized activity ($p_k(t)$) in R layer latent attractor with respect to time. Context sequences start every $n = 30$ time steps ($n - r = 10$). Context patterns are shown with circles and regular ones with dots. The number in each box represents the index of the latent attractor whose activity is shown. Top row always represents the activity in the correct latent attractor (indices 1 to 5). The order of the rest of the attractors on the vertical axis is such that candidate attractors earlier in the sequence are presented in upper rows, while other attractors - candidate or not - in bottom rows. Top row activity always converges to one, showing that the correct attractor is activated.

Each sequence S^q has $n = 30$ patterns, of which the first $r = 20$ represent the context sequence and the last $n-r$ are the regular sequence. The context sequence contains $l = 5$ context patterns and the remaining $r - l$ are non-context patterns. Both non-context patterns in the context sequence and patterns in the regular sequence are chosen randomly from a set R. At the beginning of each sequence, S^q, the recurrent gain for all R neurons is set to a value below stability threshold [12]. Depending on how many context groups are simultaneously stimulated by the incoming context patterns from C^q, the activity in R and H is distributed among the excited attractors. The recurrent gain of neurons in these attractors goes up, while that of other neurons decreases. At the end of a context sequence, only one attractor is consistent with the whole set of context stimuli in C^q, and almost all activity should be concentrated in its active set.

Figure 1.4 shows the result of a single network simulation, when context sequences are presented at the input. Plots represents the normalized activity within the active set of an attractor α in R layer:

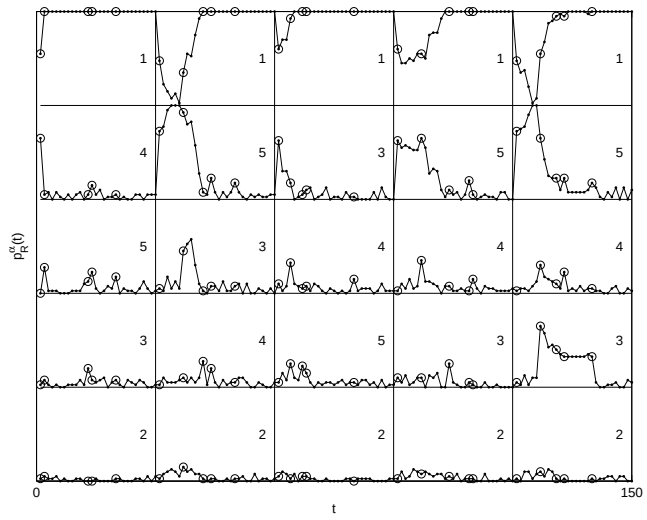

Figure 1.5. Five repeats of the same context sequence (Figure 1.4, leftmost column), with different context pattern order each time. A context sequence ($r = 20$) is followed by a regular sequence of $n - r = 10$ patterns. The order of the latent attractors has the same significance as in Figure 1.4.

$p_R^\alpha(t) = a_R^\alpha(t)/K_R$. It can be seen that, for each context sequence, the activity in only one of the attractors goes up steadily. In all other attractors the activity might increase for a few time steps, but it finally shuts down. In between consecutive context patterns, the activity is spread approximately equally between the candidate attractors.

Figure 1.5 shows the results of a simulation where one context sequence is presented repeatedly for five times. Each time the order in which context patterns are presented is different. Each context sequence is followed by $n - r = 10$ regular patterns. It is clear that the activity remains confined within the chosen attractor even though the regular patterns have no association with any attractor. It can be seen that sometimes a wrong attractor almost wins the competition in the middle of a context sequence, but it is finally shut down.

In Figure 1.6, there are only two sequences, and the overlap between them is varied as follows: Plot (a). The first two context patterns overlap; Plot (b). The first three context patterns overlap. The order between context patterns is kept the same and the interval between consecutive context patterns is constant. Figure 1.6 shows the mean normalized activity ($< p_R(t) >$) in two attractors as compared to the mean activity in non-context attractors. For an overlap of two patterns, (Figure 1.6 (a)), as well as for an overlap of three patterns (Figure 1.6 (b)), the mean activity in the context attractors goes up slowly, and in

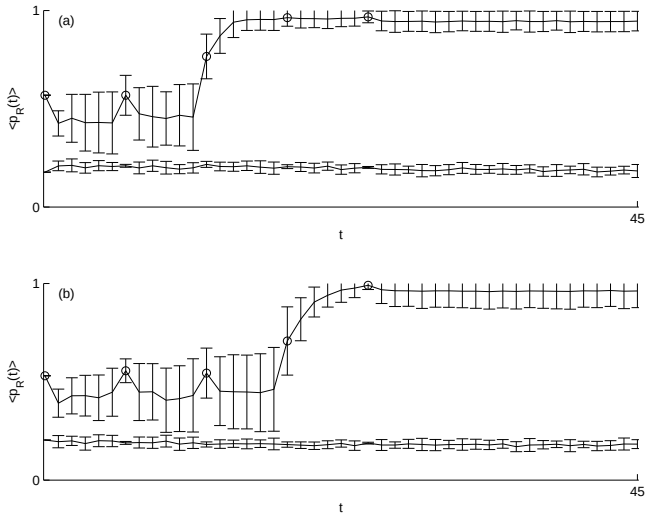

Figure 1.6. Each graph shows the mean normalized activity level ($< p_R(t) >$) in two context attractors (upper curve) and the mean activity level in non-context attractors (lower curve) during the presentation of a stimulus sequence. The mean is taken over five different networks and five different presentations of the same context patterns. In plot (a) there is an overlap of two context patterns between the two selected attractors (first two context patterns), while in Figure (b) there is an overlap of three context patterns (first three). Each context sequence ($r = 25, l = 5$) has a constant inter-context pattern interval of five regular patterns, and is followed by $n - r = 20$ regular patterns.

between context patterns it remains almost at the same level. Eventually, the correct attractor is selected in both cases.

Figure 1.7 shows a sample simulation result obtained with one network. The normalized activity levels in two attractors is shown as it varies in time. In the first half of the time, the context patterns of one attractor are presented to the network input, while in the second half, the context patterns of the second attractor. The first two context patterns are shared between the two attractors. The activity levels during the presentation of the ambiguous patterns and of the interspersed non-relevant patterns tend to remain low and approximately equally spread between the two attractors.

8. Simulation Results for First-Order Multi-Scale Context Dependence

The model shown in Figure 1.3 was simulated with the following parameter values: $N_E = N_D = 20$, $N_C = 500$, $K_C = 150$, $N_R = 1000$,

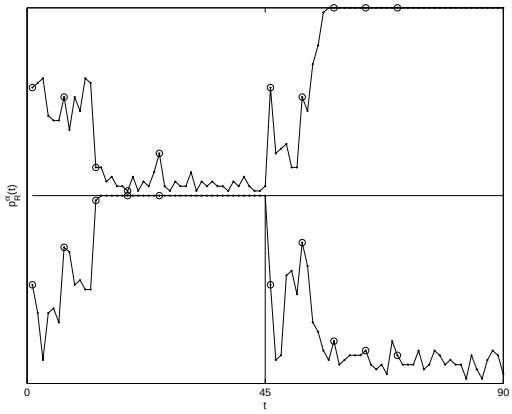

Figure 1.7. The two graphs represent the normalized activity in two attractors, where the first two context patterns in both context sequences are identical. Presentation of context patterns is marked by circles. There are $r = 5$ regular patterns between context ones, and each context sequence is followed by $n - r = 20$ regular patterns.

$G_R = 100$, $K_R = 40$, $N_H = 500$, $G_H = 50$, $K_H = 45$, $N_P = 1000$, $K_P = 40$, $\theta = 1.1$, $p_{DR} = 0.4$, $p_{RD} = 0.9$, $p_{RH} = 0.7$, $p_{HR} = 0.9$, $p_{RP} = 0.04$, $p_{PD} = 0.9$, $\tau_f = 3$. There were $M = 5$ attractors stored in the connections within the LA module. Initial weight values were chosen in intervals as follows: $w_{CR} \in (0.2,\ 0.4)$, $w_{DR} \in (0.4, 0.8)$, $w_{RD} \in (0.05,\ 0.08)$, $w_{RP} \in (0.4,\ 0.8)$, $w_{PD} \in (0.01,\ 0.05)$. Maximum weight values during learning were: $w_{max}^{RD} = 0.25$, $w_{max}^{PD} = 1$. Connection gains were: $g_{CR} = 1$, $g_{ED} = 1$, $g_{DD} = 0.5$, $g_{RD} = 0.09$, $g_{DR} = 0.9$, $g_{RH} = 1$, $g_{HR} = 0.04$, $g_{RP} = 1$, $g_{PD} = 0.15$. Learning parameters were selected as follows: for fast LTP: $\alpha_{min} = 0.015$ and $\alpha_{max} = 0.01$, for fast LTD: $\alpha_{min} = \alpha_{max} = 0.001$, for slow LTP: $\alpha_{min} = 0.05$ and $\alpha_{max} = 0.03$, for fast LTD: $\alpha_{min} = 0.003$ and $\alpha_{max} = 0.001$.

Simulations were done with a V set of $N_V = 20$ patterns, from where $P = 4$ canonical sequences were selected, each containing $l = 4$ canonical patterns and episodes with $n = 10$ patterns.

Figure 8 shows the recall quality $Q_{recall} = (1/N_r P) \sum_{p=1}^{P} (N_{correct}/l)$ over time. Every five steps, the system was tested in recall mode for $N_r = 5$ times and Q_{recall} was measured. N_{recall} is the number of correct firings in D layer as predicted by the P layer.

Figure 1.8 shows the mean input over canonical D neurons and over distractor D neurons. At $t = 0$, D neurons for both distractor and canonical patterns receive the same R input, but over time, the R projection differentiates, increasing for canonical neurons and decreasing for distractor neurons.

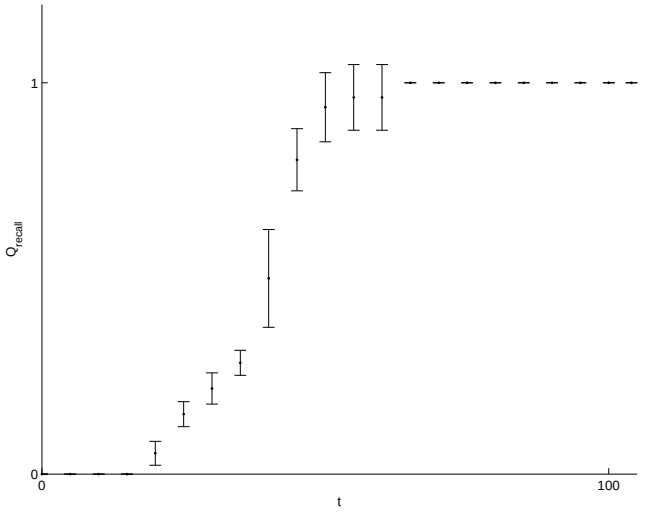

Figure 1.8. Mean and standard deviation of Q_{recall} averaged over five different networks. Every five steps in learning mode, Q_{recall} was assessed by testing the network in recall mode repeated five times.

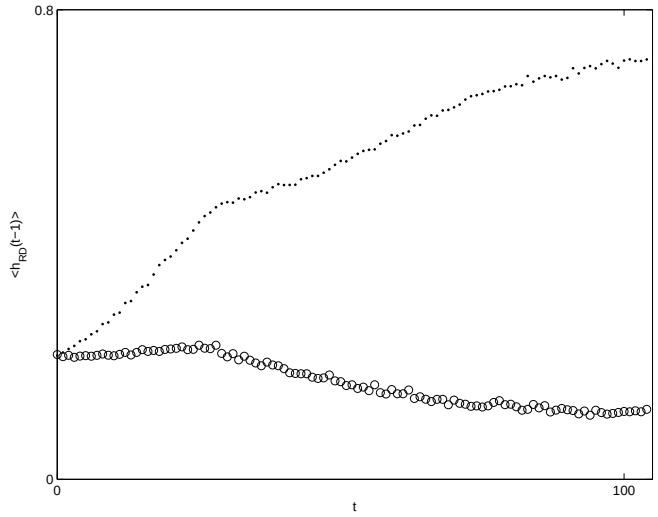

Figure 1.9. Mean R input at $t-1$ over D layer neurons averaged over P sequences and over five networks. Dots indicate the mean R input over canonical D neurons, and circles the mean R input over distractor D neurons.

In the second set of simulations, all canonical sequences contained the same stimuli, but in a different order. This is an especially difficult problem because the sequences are differentiated *only* by pattern order

and not by identity. The system – with the same set of parameters – was able to generate all canonical sequences almost perfectly. When it did encounter problems recalling the right order, a subsequence was recalled repeatedly. For example, if the canonical sequence for one episode was: $S^1 = 16, 6, 3, 1$, the system recalled either the correct sequence or one of the following: $1, 6, 3, 1$, or $6, 3, 1, 6$. In both cases, it recalled the last three patterns correctly, but it had problems generating the first canonical pattern. Figure 1.9 (a) and Figure 1.10 (a) show the Q_{recall} and mean R input for this situation.

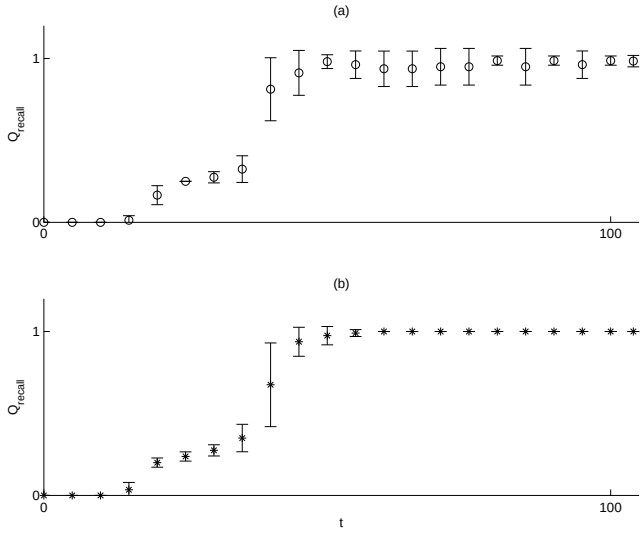

Figure 1.10. Mean and standard deviation of Q_{recall} averaged over five different networks. Every five steps in learning mode, Q_{recall} was assessed by testing the network in recall mode repeated five times.

For the last set of simulations, all canonical sequences were the same – the same canonical patterns in the same order – but with different contexts. Even in this extreme case, the system - with no change in parameter values - was able to generate perfectly the same sequence in different contexts. This shows that the LA module generated context-dependent representations of the same stimuli, and through the P layer these distinct representations were associated with the same D neuron. Figure 1.9 (b) and Figure 1.10 (b) show the results of these simulations.

In other experiments not shown here, we varied the number of canonical patterns and the number of distractors. As long as the probability of distractors is kept relatively low compared to that of canonical stimuli, the system has no difficulty in learning and recalling sequences.

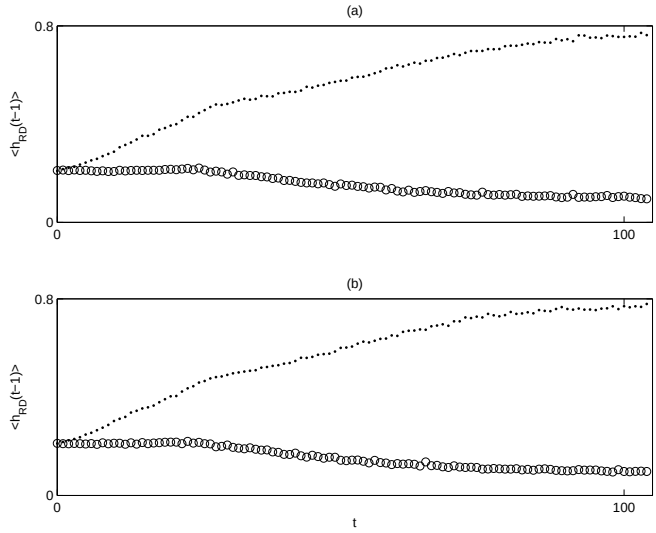

Figure 1.11. Mean R input at $t-1$ over D layer neurons averaged over P sequences and over five networks. Dots indicate the mean R input over canonical D neurons, and circles the mean R input over distractor D neurons.

9. Latent Attractors as a General Paradigm

The focus in this chapter has been on demonstrating that latent attractors are a general and flexible way for representing complex context dependencies in connectionist systems. However, the utility of latent attractors goes well beyond this; they represent a generalization of the well-established paradigm of attractor networks [1, 2, 32, 33].

Attractor-based computation has been a mainstay of neural information processing from the field's inception. The notion of using stable patterns of activity as units of information is a fundamental insight that enables a profoundly rich mode of computation: Computing with dynamical objects. It has been suggested that all cognition (both perception and action) can ultimately be modeled in this way [28, 37, 76, 64, 77], and this idea is also implicit in the Gestalt approach [26]. While most of the focus has been on stable patterns [32, 33, 38] or cyclical attractors [75, 35, 81, 58], there have been very interesting models based on chaotic dynamics [25, 3, 39] and on interacting attractors [8, 55, 54, 43]. Latent attractors, as described above, add a new dimension of possibilities in this regard. They can be seen broadly as a general mechanism for incorporating *dynamic soft modularity* into networks.

Most attractor networks are homogeneous and monolithic, i.e., every computation (e.g., memory retrieval) is performed over the entire

network. However, this does not have to be the case, and several researchers have proposed networks comprising multiple hierarchically arranged modules [19, 21, 8, 59, 55, 54, 43, 18]. One obvious benefit offered by such modular networks is that, in principle, computation can use the modules combinatorially to encode a very large number of patterns. If each module produces dynamic rather than static patterns (e.g., sequences), a very large repertoire of dynamical behaviors can be produced by using different combinations of just a few modules. Indeed, evolution seems to do just this in using specific groups of genes (e.g., homeotic genes) for different purposes in different organisms and at different developmental stages within the same organism [83, 7]. Latent attractors can support a very flexible version of such modularity where the modules, rather than being fixed and non-overlapping are, in fact, adaptive, transient and overlapping. They can be switched on and off by afferent or re-entrant stimuli, creating transient "soft-assembled" networks to produce specific behaviors in particular contexts. Furthermore, because the attractors are latent, they leave the selected neurons free to exhibit dynamic behavior within the constraints of the bias mask. In principle, therefore, a network with multiple, interacting layers of latent attractors could produce a vast range of dynamical behaviors in a flexible yet controllable way. We will present results for such systems in future reports.

10. Conclusion

In this chapter, we have presented latent attractors as a flexible and general paradigm for complex spatiotemporal computation in neural systems. We have demonstrated the utility of the approach in the case of several complex context-dependent tasks, and argued for a broader range of possible applications.

Acknowledgments

The authors wish to thank Phil Best, Mike Hasselmo, Chip Levy, John Lisman, David Redish, Bill Skaggs, Dave Touretzky and DeLiang Wang for stimulating ideas. This material is partially based on work supported by the National Science Foundation under grants IBN-9634424 and IBN-9808664.

References

[1] S. Amari and K. Maginu. Statistical neurodynamics of associative memory. *Neural Networks*, 1:63–73, 1988.

[2] D.J. Amit. *Modeling brain function: the world of attractor neural networks*. Cambridge University Press, 1989.

[3] A. Babloyantz and A. Desthexhe. Low-dimensional chaos in an instance of epilepsy. *Proceedings of the National Academy of Sciences USA*, 83:3513–3517, 1986.

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

[5] M. Botvinick and D.C. Plaut. Doing without schema hierarchies: A recurrent connectionist approach to normal and impaired routine sequential action. *Psychological Review*, 111:395–429, 2004.

[6] N. Burgess. Neuronal computations underlying the firing of place cells and their role in navigation. *Hippocampus*, 6:749–762, 1996.

[7] S.B. Carroll. Endless forms: The evolution of gene regulation and morphological diversity. *Cell*, 101:577–580, 2000.

[8] C. Cortes, A. Krogh, and J.A Hertz. Hierarchical associative memories. *Journal of Physics A*, 20:4449–4455, 1987.

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

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

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

[12] S. Doboli and A.A Minai. Network capacity analysis for latent attractor computation. *Network: Computation in Neural Systems*, 14:273–302, 2003.

[13] S. Doboli and A.A Minai. Using latent attractors to discern temporal order. In *Proceedings of IJCNN'04, Budapest, Hungary*, 2004.

[14] S. Doboli, A.A. Minai, and P.J. Best. A comparison of context-dependent hippocampal place codes in 1-layer and 2-layer recurrent neural networks. In *Proceedings of the 1999 Computational Neuroscience Conference (CNS'99)*, 1999.

[15] S. Doboli, A.A. Minai, and P.J. Best. Generating smooth context-dependent representations. In *Proceedings of IJCNN'99, Washington D.C.*, 1999.

[16] 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.

[17] 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.

[18] D.R.C. Dominguez. Information capacity of a hierarchical neural network. *Physical Review E*, 58:4811–4815, 1998.

[19] V.S. Dostenko. Hierarchical model of memory. *Physica A*, 140:410–415, 1986.

[20] J. Duncan. An adaptive coding model pf neural function in prefrontal cortex. *Nature Reviews*, 1:59–65, 2000.

[21] G.M. Edelman. *Neural Darwinism: The theory of neuronal group selection.* Basic Books, 1987.

[22] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

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

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

[25] W.J. Freeman. Tutorial on neurobuiology: From single neurons to brain chaos. *International Journal of Bifurcation and Chaos*, 2:451–482, 1992.

[26] J.J. Gibson. *The ecological approach to visual perception.* Houghton Mifflin, 1979.

[27] 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.

[28] H. Haken, J.A.S. Kelso, and H. Bunz. A theoretical model of phase transition in human hand movement. *Biological Cybernetics*, 51:347–356, 1985.

[29] Z.-S. Han, E.H. Buhl, Z. Lörinczi, and P. Somogyi. A high degree of spatial selectivity in the axonal and dendritic domains of physiologically identified local-circuit neurons in the dentate gyrus of the rat hippocampus. *European Journal of Neuroscience*, 5:395–410, 1993.

[30] M.E. Hasselmo, E. Schnell, and E. Barkai. Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation

in hippocampal region CA3. *Journal of Neuroscience*, 15:5249–5262, 1995.

[31] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[32] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, 79:2554–2558, 1982.

[33] J.J. Hopfield. Neurons with graded response have collective computationasl properties like those of two-state neurons. *Proceedings of the National Academy of Sciences USA*, 81:3088–3092, 1984.

[34] M.B. Jackson and H.E. Scharfman. Positive feedback from hilar mossy cells to granule cells in the dentate gyrus revealed by voltage-sensitive dye and microelectrode recording. *Journal of Neurophysiology*, 76:601–616, 1996.

[35] 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.

[36] S.A. Kauffman. *The Origins of Order*. Oxford University Press, 1993.

[37] J.A.S. Kelso. *Dynamic patterns: The self-organization of brain and behavior*. MIT Press, 1995.

[38] B. Kosko. Bidirectional associative memories. *IEEE Transactions on Systems, Man, and Cybernetics*, 18:49–60, 1988.

[39] R. Kozma and W.J. Freeman. Chaotic resonance - methods and applications for robust classification of noisy and variable patterns. *International Journal of Bifurcation and Chaos*, 11:1607–1629, 2001.

[40] W.B. Levy. A computational approach to hippocampal function. In R.D. Hawkins and G.H. Bower, editors, *Computational Models of Learning in Simple Neural Systems*, pages 243–305. Academic Press, San Diego, CA, 1989.

[41] W.B. Levy. A sequence predicting CA3 is a flexible associator that learns and uses context to solve hippocampal-like tasks. *Hippocampus*, 6:579–591, 1996.

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

[43] W.E. Lillo, D.C. Miller, S. Hui, and S.H. Zak. Synthesis of brain-state-in-a-box (bsb) based associative memories. *IEEE Transactions on Neural Networks*, 5:730–737, 1994.

[44] J.E. Lisman. Relating hippocampal circuitry to function: The role of reciprocal dentate-CA3 interaction in the recall of sequences. *Neuron*, 22:233–242, 1999.

[45] E.J. Markus, Y.-L. Qin, B. Leonard, W.E. Skaggs, B.L. McNaughton, and C.A. Barnes. Interactions between location and task affect the spatial and directional firing of hippocampal neurons. *Journal of Neuroscience*, 15:7079–7094, 1995.

[46] D. Marr. Simple memory: A theory for archicortex. *Philosophical Transactions of the Royal Society of London B*, 262:23–81, 1971.

[47] M. R. Mehta. Cooperative ltp can map memory sequences on dendritic branches. *Trends in Neurosciences*, 27(2):69–72, 2004.

[48] E.K. Miller. The prefrontal cortex and cognitive control. *Nature Reviews*, 2:820–829, 2001.

[49] A.A. Minai, G.L. Barrows, and W.B. Levy. Disambiguation of pattern sequences with recurrent networks. In *Proceedings of the 1994 World Congress on Neural Networks, San Diego, CA*, volume IV, pages 176–180, 1994.

[50] A.A. Minai and P.J. Best. Encoding spatial context: A hypothesis on the function of the dentate gyrus-hilus system. In *Proceedings of the 1998 International Joint Conference on Neural Networks, Anchorage, AK*, pages 587–592, 1998.

[51] E.I. Moser. Altered inhibition of dentate granule cells during spatial learning in an exploration task. *Journal of Neuroscience*, 16:1247–1259, 1996.

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

[53] M.C. Mozer. Induction of multiscale temporal structure. *Advances in Neural Information Processing Systems 4*, pages 275–282, 1992.

[54] D. O'Kane and D. Sherrington. A feature retrieving attractor neural network. *Journal of Physics A*, 26:2333–2342, 1993.

[55] D. O'Kane and A. Treves. Short- and long-range connections in autoassociative memory. *Journal of Physics A*, 25:5055–5069, 1992.

[56] J. O'Keefe and L. Nadel. *The Hippocampus as a Cognitive Map*. Clarendon Press, Oxford, UK, 1978.

[57] R.C O'Reilly and M.J. Frank. Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, 18(2):283–328, 2005.

[58] B.A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1:263–269, 1989.

[59] M.A. Pires Idiart and A. Theumann. Retrieval properties of neural networks with hierarchical clustering. *Journal of Physics A*, 24:L649–L658, 1991.

[60] G.J. Quirk, R.U. Muller, and J.L. Kubie. The firing of hippocampal place cells in the dark depends on the rat's recent experience. *Journal of Neuroscience*, 10:2008–2017, 1990.

[61] A.D. Redish. *Beyond the Cognitive Map: Contributions to a Computational Neuroscience Theory of Rodent Navigation*. PhD thesis, Carnegie-Mellon University, 1997.

[62] A.D. Redish and D.S. Touretzky. Cognitive maps beyond the hippocampus. *Hippocampus*, 7:15–35, 1997.

[63] M. Reiss and J.G. Taylor. Storing temporal sequences. *Neural Networks*, 4:773–787, 1991.

[64] M.A. Riley and M.T. Turvey. Variability and determinism in motor behavior. *Journal of Motor Behavior*, 34:99–125, 2002.

[65] E.T. Rolls. The representation and storage of information in neuronal networks in the primate cerebral cortex and hippocampus. In R. Durbin, C. Miall, and G. Mitchison, editors, *The Computing Neuron*, pages 125–159. Addison-Wesley, Reading, MA, 1989.

[66] A. Rotenberg and R.U. Muller. Variable place-cell coupling to a continuously viewed stimulus: Evidence that the hippocampus acts as a perceptual system. *Philosophical Transactions of the Royal Society of London B*, 352:1505–1513, 1997.

[67] N.P. Rougier, D.C. Noelle, T.S. Braver, J.D. Cohen, and R.C. O'Reilly. Prefrontal cortex and flexible cognitive control: Rules without symbols. *Proceedings of the National Academy of Sciences USA*, 102:7338–7343, 2005.

[68] N.P. Rougier and R.C. O'Reilly. Learning representations in a gated prefrontal cortex model of dynamic task switching. *Cognitive Science*, 26:503–520, 2002.

[69] A. Samsonovich and B.L. McNaughton. Path integration and cognitive mapping in a continuous attractor neural network model. *Journal of Neuroscience*, 17:5900–5920, 1997.

[70] J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4:131–139, 1992.

[71] D. Servan-Schreiber, A. Cleereman, and J.L. McClelland. Learning sequential structure in simple recurrent networks. *Advances in Neural Information Processing Systems 1*, pages 643–652, 1989.

[72] I. Shmulevich, E.R. Dougherty, and W. Zhang. From boolean to probabilistic boolean networks as models of genetic regularoty networks. *Proceedings of the IEEE*, 90:1778–1792, 2002.

[73] W.E. Skaggs and B.L. McNaughton. Replay of neuronal firing sequences in rat hippocampus during sleep following spatial experience. *Science*, 271:1870–1873, 1996.

[74] V.S. Sohal and M.E. Hasselmo. $GABA_B$ modulation improves sequence disambiguation in computational models of hippocampal region CA3. *Hippocampus*, 8:171–193, 1998.

[75] H. Sompolinsky and I. Kanter. Temporal association in asymmetric neural networks. *Physical Review Letters*, 57:2861–2864, 1986.

[76] M.T. Turvey. Coordination. *American Psychologist*, 45:938–953, 1990.

[77] G.C. Van Orden, J.G. Holden, and M.T. Turvey. Self-organization of cognitive performance. *Journal of Experimental Psychology: General*, 132:331–350, 2003.

[78] D.L. Wang and B. Yuwono. Anticipation-based temporal pattern generation. *IEEE Transactions on Systems, Man, and Cybernetics*, 25:615–628, 1995.

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

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

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

[82] D. Willshaw, O.P. Buneman, and H.C. Longuet-Higgins. Nonholographic associative memory. *Nature*, 222:960–962, 1969.

[83] L. Wolpert, R. Beddington, J. Brockes, T. Jessell, P. Lawrence, and E. Meyerowitz. *Principles of Development*. Oxford University Press, 1998.

[84] 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.