1 What makes a dynamical system computationally powerful?

Robert Legenstein and Wolfgang Maass
Institute for Theoretical Computer Science
Technische Universität Graz
Austria, 8010 Graz
{*legi, maass*}*@igi.tugraz.at*

We review methods for estimating the computational capability of a complex dynamical system. The main examples that we discuss are models for cortical neural microcircuits with varying degrees of biological accuracy, in the context of online computations on complex input streams. We address in particular the question to what extent earlier results about the relationship between the edge of chaos and the computational power of dynamical systems in discrete time for off-line computing also apply to this case.

1.1 Introduction

Most work in the theory of computations in circuits focuses on computations in feedforward circuits, probably because computations in feedforward circuits are much easier to analyze. But biological neural circuits are obviously recurrent, in fact the existence of feedback connections on several spatial scales is a characteristic property of the brain. Therefore an alternative computational theory had to be developed for this case. One neuroscientist who emphasized the need to analyze information processing in the brain in the context of dynamical systems theory was Walter Freeman, who started to write a number of influential papers on this topic in the 1960s; see (Freeman, 1975) and (Freeman, 2000) for references and more recent accounts. The theoretical investigation of computational properties of recurrent neural circuits started shortly afterwards. Earlier work focused on the engraving of attractors into such systems in order to restrict the dynamics to achieve well-defined properties. One stream of work in this direction (see, e.g. Grossberg, 1967;

D., 1968; Amari, 1972; Little, 1974) culminated in the influential studies of Hopfield regarding networks with stable memory, called Hopfield networks (Hopfield, 1982, 1984), and the work of Hopfield and Tank on networks which are able to find approximate solutions of hard combinatorial problems like the traveling salesman problem (Hopfield and Tank, 1985, 1986). The Hopfield network is a fully connected neural networks of threshold or threshold-like elements. Such networks exhibit rich dynamics and are chaotic in general. However, Hopfield assumed symmetric weights, which strongly constrains the dynamics of the system. Specifically, one can show that only point attractors can emerge in the dynamics of the system, i.e. the activity of the elements always evolves to one of a set of stable states which is then kept forever.

Somewhat later the alternative idea arose to use the rich dynamics of neural systems which can be observed in cortical circuits rather than to restrict them (Buonomano and Merzenich, 1995). In addition one realized that one needs to look at online computations (rather than off-line or batch computing) in dynamical systems in order to capture the biologically relevant case (see Maass and Markram, 2005, for definitions of such basic concepts of computation theory). These efforts resulted in the "liquid state machine" model by Maass et al. (2002) and the "echo state network" by Jäger (2002) which were introduced independently. The basic idea of these models is to use a recurrent network to hold and nonlinearly transform information about the past input stream in the high-dimensional transient state of the network. This information can then be used to produce in real-time various desired online outputs by simple linear readout elements. These readouts can be trained to recognize common information in dynamical changing network states because of the high dimensionality of these states. It has been shown that these models exhibit high computational power (Legenstein et al., 2003; Joshi and Maass, 2004; Jäger and Haas, 2004). However, the analytical study of such networks with rich dynamics is a hard job. Fortunately, there exists a vast body of literature on related questions in the context of many different scientific disciplines in the more general framework of dynamical systems theory. Specifically, a stream of research is concerned with system dynamics located at the boundary region between ordered and chaotic behavior which was termed "edge of chaos". This research is of special interest for the study of for neural systems because it was shown that the behavior of dynamical systems is most interesting in this region. Furthermore, a link between computational power of dynamical systems and the edge of chaos was conjectured.

It is therefore a promising goal to use concepts and methods from dynamical systems theory to analyze neural circuits with rich dynamics and to get in this way better tools for understanding computation in the brain. In this chapter, we will take a tour, visiting research concerned with the edge of chaos and eventually arrive at a first step towards this goal. The aim of this chapter is to guide the reader through a stream of ideas which we believe are inspiring for research in neuroscience and molecular biology, as well as for the design of novel computational devices in engineering.

After a brief introduction of fundamental principles of dynamical system and

Table 1.1 General properties of various types of dynamical systems.

	cellular automata	iterative maps	Boolean circuits	cortical microcircuits and gene regulation networks
analog states?	no	yes	no	yes
continuous time?	no	no	no	yes
high-dimensional?	yes	no	yes	yes
with noise?	no	no	no	yes
with online input?	no	no	usually no	yes

chaos in Section 1.2, we will start our journey in Section 1.3 in the field of theoretical biology. There, Kauffman studied questions of evolution and emerging order in organisms. We will see that depending on the connectivity structure, networks may operate either in an ordered or chaotic regime. Furthermore, we will encounter the edge of chaos as a transition between these dynamic regimes. In Section 1.4, our tour will visit the field of statistical physics, where Derrida and others studied related questions and provided new methods for their mathematical analysis. In Section 1.5 the reader will see how these ideas can be applied in the theory of computation. The study of cellular automata by Wolfram, Langton, Packard, and others led to the conjecture that complex computation are best performed in systems at the edge of chaos. The next stops of our journey in Section 1.6 and Section 1.7 will bring us close to our goal. We will review work by Bertschinger and Natschläger who analyzed real-time computations on the edge of chaos in threshold circuits. In Section 1.8, we will briefly examine self-organized criticality, i.e. how a system can adapt its own dynamics towards the edge of chaos. Finally, Section 1.9 presents the efforts of the authors of this chapter to apply these ideas to computational questions in the context of biologically realistic neural microcircuit models. In this section we will analyze the edge of chaos in networks of spiking neurons and ask the following question: In what dynamical regimes are neural microcircuits computationally most powerful? Table 1 shows that neural microcircuits (as well as gene regulation networks) differ in several essential aspects from those examples for dynamical systems that are commonly studied in dynamical systems theory.

1.2 Chaos in Dynamical Systems

In this section we briefly introduce ideas from dynamical systems theory and chaos. A few slightly different definitions of chaos are given in the literature. Although we will mostly deal here with systems in discrete time and discrete state space, we start out with the well established definition of chaos in continuous systems and return to discrete systems later in this section.

The subject known as *dynamics* deals with systems that evolve in time (Strogatz,

1994). The system in question may settle down to an equilibrium, may enter a periodic trajectory (limit cycle), or do something more complicated. In (Kaplan and Glass, 1995) the dynamics of a deterministic system is defined as being chaotic if it is aperiodic and bounded with sensitive dependence on initial conditions.

The phase space for an N-dimensional system is the space with coordinates $x_1, \ldots, x_N$. The state of an N-dimensional dynamical system at time t is represented by the state vector $\mathbf{x}(t) = (x_1(t), \ldots, x_N(t))$. If a system starts in some initial condition $\mathbf{x}(0)$, it will evolve according to its dynamics and describe a trajectory in state space. A steady state of the system is a state $\mathbf{x}_s$ such that if the system evolves with $\mathbf{x}_s$ as its initial state, it will remain in this state for all future times. Steady states may or may not be stable to small outside perturbations. For a stable steady state, small perturbations die out and the trajectory converges back to the steady state. For an unstable steady state, trajectories do not converge back to the steady state after arbitrarily small perturbations.

The general definition of an *attractor* is a set of points or states in state space to which trajectories within some volume of state space converge asymptotically over time. This set itself is invariant under the dynamic evolution of the system.[1] Therefore, a stable steady state is a zero-dimensional, or point, attractor. The set of initial conditions that evolve to an attractor A is called the *basin of attraction* of A. A *limit cycle* is an isolated closed trajectory. Isolated means that neighboring trajectories are not closed. If released in some point of the limit cycle, the system flows on the cycle repeatedly. The limit cycle is stable if all neighboring trajectories approach the limit cycle. A stable limit cycle is a simple type of an attractor. Higher dimensional and more complex types of attractors exist.

In addition, there exist also so called *strange*, or *chaotic attractors*. For example all trajectories in a high dimensional state space might be brought onto a two-dimensional surface of some manifold. The interesting property of such attractors is that, if the system is released in two different experiments from two points on the attractor which are arbitrarily close to each other, the subsequent trajectories remain on the attractor surface but diverge away from each other. After a sufficient time, the two trajectories can be arbitrarily far apart from each other. This extreme sensitivity to initial conditions is characteristic for chaotic behavior. In fact, exponentially fast divergence of trajectories (characterized by a positive Lyapunov exponent) is often used as a definition of chaotic dynamics (see e.g. Kantz and

1. For the sake of completeness, we give here the definition of an attractor according to Strogatz (1994): He defines an attractor to be a closed set A with the following properties:

a A is an *invariant set*: any trajectory $\mathbf{x}(t)$ that starts in A stays in A for all time.

b A attracts an open set of initial conditions: there is an open set U containing A such that if $\mathbf{x}(t) \in U$, then the distance from $\mathbf{x}(t)$ to A tends to zero as $t \to \infty$. The largest such U is called the *basin of attraction* of A.

c A is *minimal*: there is no proper subset of A that satisfies conditions a and b.

Schreiber, 1997). There might be a lot of structure in chaotic dynamics since the trajectory of a high-dimensional system might be projected merely onto a two-dimensional surface. However, since the trajectory on the attractor is chaotic, the exact trajectory is practically not predictable (even if the system is deterministic).

Systems in discrete time and with a finite discrete state space differ from continuous systems in in several aspects. First, since state variables are discrete, trajectories can merge, whereas in a continuous system they may merely approximate each other. Second, since there is a finite number of states, the system must eventually reenter a state previously encountered and will thereafter cycle repeatedly through this state cycle. These state cycles are the dynamical attractors of the discrete system. The set of states flowing into such state cycle or lying on it constitutes the basin of attraction of that state cycle. The length of a state cycle is the number of states on the cycle. For example, the memory states in a Hopfield network (a network of artificial neurons with symmetric weights) are the stable states of the system. A Hopfield network does not have state cycles of length larger then one. The basins of attraction of memory states are used to drive the system from related initial conditions to the same memory state, hence constituting an associative memory device.

Characteristic properties of chaotic behavior in discrete systems are a large length of state cycles and high sensitivity to initial conditions. Ordered networks have short state cycles and their sensitivity to initial conditions is low, i.e. state cycles are quite stable. We note that state cycles can be stable with respect to some small perturbations, but unstable to others. Therefore, "quite stable" means in this context that the state cycle is stable to a high percentage of small perturbations. These general definitions are not very precise and will be made more specific for each of the subsequent concrete examples.

1.3 Randomly Connected Boolean Networks

The study of complex systems is obviously important in many scientific areas. In genetic regulatory networks, thousands or millions of coupled variables orchestrate developmental programs of an organism. In 1969, Kauffman started to study such systems in the simplified model of Boolean networks (Kauffman, 1969, 1993). He discovered some surprising results which will be discussed in this section. We will encounter systems in the ordered and in the chaotic phase. The specific phase depends on some simple structural feature of the system and a phase transition will occur when this feature changes.

A Boolean network consists of N elements and connections between them. The state of its elements is described by binary variables $x_1, \ldots, x_N$. The dynamical behavior of each variable, whether it will be active (1) or inactive (-1) at the next

time step, is governed by a Boolean function.[2] The (directed) connections between the elements describe possible interactions. If there is a connection from element i to element j, then the state of element i influences the state of element j in the next time step. We say that i is an input of element j.

An initial condition is given by a value for each variable $\mathbf{x}(0)$. Thereafter, the state of each element evolves according to the Boolean function assigned to it. We can describe the dynamics of the system by a set of iterated maps

$$\begin{aligned} x_1(t+1) &= f_1(x_1(t), \ldots, x_N(t)) \\ &\vdots \\ x_N(t+1) &= f_N(x_1(t), \ldots, x_N(t)), \end{aligned}$$

where $f_1, \ldots, f_N$ are Boolean functions.[3] Here, all state variables are updated in parallel at each time step.

The stability of attractors in Boolean networks can be studied with respect to minimal perturbations. A minimal perturbation is just the flip of the activity of a single variable to the opposite state.

Kauffman studied the dynamics of Boolean networks as a function of the number of elements in the network N, and the average number K of inputs to each element in the net. Since he was not interested in the behavior of particular nets but rather in the expected dynamics of nets with some given N and K, he sampled at random from the ensemble of all such networks. Thus the K inputs to each element were first chosen at random and then fixed, and the Boolean function assigned to each element was also chosen at random and then fixed. For each such member of the ensemble Kauffman performed computer simulations and examined the accumulated statistics.

The case $K = N$ is especially easy to analyze. Since the Boolean function of each element was chosen randomly from a uniform distribution, the successor to each circuit state is drawn randomly from a uniform distribution among the 2^N possible states. This leads to long state cycles. The median state cycle length is $0.5 \cdot 2^{N/2}$. Kauffman called such exponentially long state cycles *chaotic*.[4] These state cycles are unstable to most perturbations, hence there is a strong dependence on initial conditions. However, only a few different state cycles exit in this case: the expected number of state cycles is N/e. Therefore, there is some characteristic structure in the chaotic behavior in the sense that the system will end up in one of only a few long term behaviors.

As long as K is not too small, say $K \geq 5$, the main features of the case $K = N$

2. In (Kauffman, 1969), the inactive state of a variable is denoted by 0. We use -1 here for reasons of notational consistency.
3. Here, x_i potentially depends on all other variables $x_1, \ldots, x_N$. The function f_i can always be restricted such that x_i is determined by the inputs to elements i only.
4. In (Kauffman, 1993), a state cycle is also called attractor. Because such state cycles can be unstable to most minimal perturbations, we will avoid the term attractor here.

persist. The dynamics is still governed by relatively few state cycles of exponential length, whose expected number is at most linear in N. For $K \geq 5$, these results can be derived analytically by a rough mean field approximation. For smaller K, the approximation becomes inaccurate. However, simulations confirm that exponential state cycle length and a linear number of state cycles are characteristic for random Boolean networks with $K \geq 3$. Furthermore, these systems show high sensitivity to initial conditions (Kauffman, 1993).

The case $K = 2$ is of special interest. There, a phase transition from ordered to chaotic dynamics occurs. Numerical simulations of these systems have revealed the following characteristic features of random Boolean networks with $K = 2$ (Kauffman, 1969). The expected median state cycle length is about $\sqrt{N}$. Thus, random Boolean networks with $K = 2$ often confine their dynamical behavior to tiny subvolumes of their state space, a strong sign of order. A more detailed analysis shows that most state cycles are short, whereas there are a few long ones. The number of state cycles is about $\sqrt{N}$ and they are inherently stable to about 80 to 90 percent of all minimal transient perturbations. Hence, the state cycles of the system have large basins of attraction and the sensitivity to initial conditions is low. In addition to these characteristics which stand in stark contrast to networks of larger K, we want to emphasize three further features. First, typically at least 70 percent of the N elements have some fixed active or inactive state which is identical for all the existing state cycles of the Boolean network. This behavior establishes a *frozen core* of elements. The frozen core creates walls of constancy which break the system into functionally isolated islands of unfrozen elements. Thus, these islands are prevented from influencing one another. The boundary regime where the frozen core is just breaking up and interaction between the unfrozen islands becomes possible is the phase transition between order and chaos. Second, altering transiently the activity of a single element typically propagates but causes only alterations in the activity of a small fraction of the elements in the system. And third, deleting any single element or altering its Boolean function typically causes only modest changes in state cycles and transients. The latter two points ensure that "damage" of the system is small. We will further discuss this interesting case in the next section.

Networks with $K = 1$ operate in an ordered regime and are of little interest for us here.

1.4 The Annealed Approximation by Derrida and Pomeau

The phase transition from order to chaos is of special interest. As we shall see in the sections below there are reasons to believe that this dynamical regime is particularly well suited for computations. There were several attempts to understand the emerging order in random Boolean networks. In this section, we will review the approach of Derrida and Pomeau (1986). Their beautiful analysis gives an analytical answer to the question where such a transition occurs.

In the original model, the connectivity structure and the Boolean functions f_i of the elements i were chosen randomly but were then fixed. The dynamics of the network evolved according to this fixed network. In this case the randomness is *quenched* because the functions f_i and the connectivity do not change with time. Derrida and Pomeau presented a simple *annealed approximation* to this model which explains why there is a critical value K_c of K where the transition from order to chaos appears. This approximation also allowed the calculation of many properties of the model. In contrast to the quenched model, the annealed approximation randomly reassigns the connectivity and the Boolean functions of the elements *at each time step*. Although the assumption of the annealed approximation is quite drastic, it turns out that its agreement with observations in simulations of the quenched model is surprisingly good. The benefits of the annealed model will become clear below.

It was already pointed out that exponential state cycle length is an indicator of chaos. In the annealed approximation however, there are no fixed state cycles because the network is changed at every time step. Therefore, the calculations are based on the dependence on initial conditions. Consider two network states $\mathcal{C}_1, \mathcal{C}_2 \in \{-1, 1\}^N$. We define the *Hamming distance* $d(\mathcal{C}_1, \mathcal{C}_2)$ as the number of positions in which the two states are different. The question is whether two randomly chosen different initial network states eventually converge to the same pattern of activity over time. Or, stated in other words, given an initial state $\mathcal{C}_1$ which leads to a state $\mathcal{C}_1^{(t)}$ at time t and a different initial state $\mathcal{C}_2$ which leads to a state $\mathcal{C}_2^{(t)}$ at time t, will the Hamming distance $d(\mathcal{C}_1^{(t)}, \mathcal{C}_2^{(t)})$ converge to zero for large t? Derrida and Pomeau found that this is indeed the case for $K \leq 2$. For $K \geq 3$, the trajectories will diverge.

To be more precise, one wants to know the probability $P_1(m, n)$ that the distance $d(\mathcal{C}_1', \mathcal{C}_2')$ between the states at time $t = 1$ is m given that the distance $d(\mathcal{C}_1, \mathcal{C}_2)$ at time $t = 0$ was n. More generally, one wants to estimate the probability $P_t(m, n)$ that the network states $\mathcal{C}_1^{(t)}, \mathcal{C}_2^{(t)}$ obtained at time t are at distance m, given that $d(\mathcal{C}_1, \mathcal{C}_2) = n$ at time $t = 0$. It now becomes apparent why the annealed approximation is useful. In the annealed approximation, the state transition probabilities at different time steps are independent, which is not the case in the quenched model. For large N, one can introduce continuous variables $\frac{n}{N} = x$, Derrida et al. show that $P_1^{annealed}(m, n)$ for the annealed network has a peak around a value $m = Ny_1$ where y_1 is given by

$$y_1 = \frac{1 - (1 - x)^K}{2} \ . \tag{1.1}$$

Similarly, the probability $P_t^{annealed}(m, n)$ has a peak at $m = Ny_t$ with y_t given by

$$y_t = \frac{1 - (1 - y_{t-1})^K}{2} \tag{1.2}$$

for $t > 1$. The behavior of this iterative map can be visualized in the so called *Derrida plot*, see Figure 1.1. The plot shows the state distance at time $t + 1$ as a

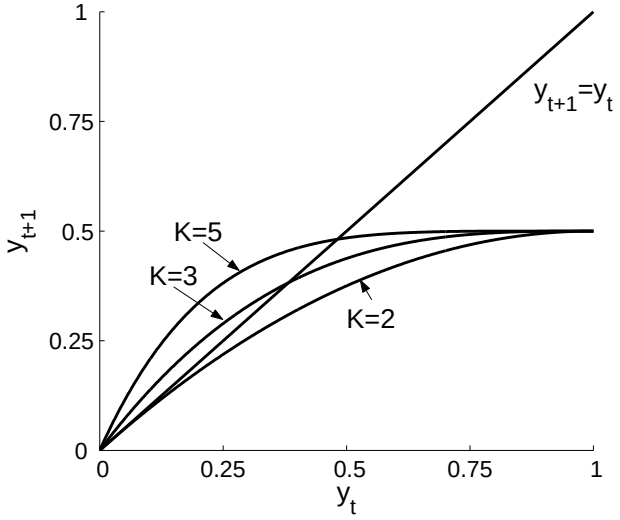

Figure 1.1 Expected distance between two states at time $t+1$ as a function of the state distance y_t between two states at time t, based on the annealed approximation. Points on the diagonal $y_t = y_{t+1}$ are fixed points of the map. The curves for $K \geq 3$ all have fixed points for a state distance larger than zero. The curve for $K = 2$ stays close to the diagonal for small state distances but does not cross it. Hence, for $K = 2$ state distances converge to zero for iterated applications of the map.

function of the state distance at time t. Points on the diagonal $y_t = y_{t+1}$ are fixed points of the map.

For $K \leq 2$, the fixed point $y = 0$ is the only fixed point of the map and it is stable. In fact, for any starting value y_1, we have $y_t \to 0$ for $t \to \infty$ in the limit of $N \to \infty$. For $K > 2$, the fixed point $y = 0$ becomes unstable and a new stable fixed point y^* appears. Therefore, the state distance need no longer always converge to zero. Hence there is a phase transition of the system at $K = 2$. The theoretical work of Derrida and Pomeau was important because before there was only empirical evidence for this phase transition.

We conclude that there exists an interesting transition region from order to chaos in these dynamical systems. For simplified models, this region can be determined analytically. In the following section we will find evidence that such phase transitions are of great interest for the computational properties of dynamical systems.

1.5 Computation at the Edge of Chaos in Cellular Automata

Evidence that systems exhibit superior computational properties near a phase transition came from the study of cellular automata. Cellular automata are quite similar to Boolean networks. The main differences are that connections between elements are local, and that an element may assume one out of k possible states

at each time step (instead of merely two states in Boolean networks). The former difference implies that there is a notion of space in a cellular automaton. More precisely, a d-dimensional space is divided into *cells* (the elements of the network). The state of a cell at time $t+1$ is a function only of its own state and the states of its immediate neighbors at time t. The latter difference is made explicit by defining a finite set Σ of cell states. The transition function Δ is a mapping from neighborhood states (including the cell itself) to the set of cell states. If the neighborhood is of size L, we have $\Delta : \Sigma^L \rightarrow \Sigma$.

What do we mean by "computation" in the context of cellular automata? In one common meaning, the transition function is interpreted as the program and the input is given by the initial state of the cellular automaton. Then, the system evolves for some specified number of time steps, or until some "goal pattern" – possibly a stable state – is reached. The final pattern is interpreted as the output of the automaton (Mitchell et al., 1993). In analogy to universal Turing machines, it has been shown that cellular automata are capable of universal computation (see e.g. von Neumann, 1966; Smith, 1971; Codd, 1968). That is, there exist cellular automata which, by getting the algorithm to be applied as part of their initial configuration, can perform any computation which is computable by any Turing machine.

In 1984, Wolfram conjectured that such powerful automata are located in a special dynamical regime. Later, Langton identified this regime to lie on a phase transition between order and chaos (see below), i.e. in the regime which corresponds to random Boolean networks with $K = 2$.

Wolfram presented a qualitative characterization of one-dimensional cellular automaton behavior where the individual automata differed by their transfer function.[5] He found evidence that all one-dimensional cellular automata fall into four distinct classes (Wolfram, 1984). The dynamics for three of these classes are shown in Figure 1.2.

- Class 1 automata evolve to a homogeneous state, i.e. a state where all cells are in the same state. Hence these systems evolve to a simple steady state.
- Class 2 automata evolve to a set of separated simple stable states or separated periodic structures of small length. These systems have short state cycles.

Both of these classes operate in the ordered regime in the sense that state cycles are short.

- Class 3 automata evolve to chaotic patterns.
- Class 4 automata have long transients, and evolve "to complex localized structures" (Wolfram, 1984).

5. He considered automata with a neighborhood of 5 cells in total and two possible cell states. Since he considered "totalistic" transfer functions only (i.e. the function depends on the *sum* of the neighborhood states only), the number of possible transfer functions was small. Hence, the behavior of all such automata could be studied.

Figure 1.2 Evolution of one-dimensional cellular automata. Each horizontal line represents one automaton state. Successive time steps are shown as successive horizontal lines. Sites with value 1 are represented by black squares; sites with value 0 by white squares. One example for an automaton of class 1 (left), class 4 (middle), and class 3 (right) is given.

Class 3 automata are operating in the chaotic regime. With chaotic, Wolfram means "unpredictability" of the exact automaton state after a few time steps. Successor states look more or less random. He also talks about non-periodic patterns. Of course these patterns are periodic if the automaton is of finite size. But in analogy with the results presented above, one can say that state cycles are very long.

Transients are the states that emerge before the dynamics reaches a stable long-lasting behavior. They appear at the beginning of the state evolution. Once the system is on a state cycle, it will never revisit such transient states. The transients of class 4 automata can be identified with large basins of attraction or high stability of state cycles. Wolfram conjectured that class 4 automata are capable of universal computations.

In 1990, Langton (1990) systematically studied the space of cellular automata considered by Wolfram with respect to an order parameter λ. This parameter λ determines a crucial property of the transfer function Δ: the fraction of entries in Δ which do not map to some prespecified quiescent state s_q. Hence, for $\lambda = 0$, all local configurations map to s_q, and the automaton state moves to a homogeneous state after one time step for every initial condition. More generally, low λ-values lead to ordered behavior. Rules with large λ tend to produce a completely different behavior.

Langton stated the following question: "Under what conditions will physical systems support the basic operations of information transmission, storage, and modification constituting the capacity to support computation?" (Langton, 1990). When Langton went through different λ values in his simulations, he found that all automaton classes of Wolfram appeared in this parametrization. Moreover, he found that the interesting class 4 automata can be found at the phase transition between ordered and chaotic behavior for λ-values between about 0.45 and 0.5, values of intermediate heterogeneity. Information theoretic analysis supported the conjectures of Wolfram, indicating that the edge of chaos is the dominant region of computationally powerful systems.

Further evidence for Wolframs hypothesis came from Packard (1988). Packard used genetic algorithms to genetically evolve one-dimensional cellular automata for

a simple computational task. The goal was to develop in this way cellular automata which behave as follows: The state of the automaton should converge to the all-one-state (i.e. the state where every cell is in state 1), if the fraction of one-states in the initial configuration is larger than 0.5. If the fraction of one-states in the initial configuration is below 0.5, it should evolve to the all-zero-state.

Mutations were accomplished by changes in the transfer function (point mutations which changed only a single entry in the rule table, and crossover which merged two rule tables into a single one). After applying a standard genetic algorithm procedure to an initial set of cellular automaton rules, he examined the rule tables of the genetically evolved automata. The majority of the evolved rule tables had λ values either around 0.23 or around 0.83. These are the two λ values where the transition from order to chaos appears for cellular automata with two states per cell.[6] "Thus, the population appears to evolve toward that part of the space of rules that marks the transition to chaos" (Packard, 1988).

These results have later been criticized (Mitchell et al., 1993). Mitchell and collaborators reexamined the ideas of Packard and performed similar simulations with a genetic algorithm. The results of these investigations differed from Packards results. The density of automata after evolution was symmetrically peaked around $\lambda = 0.5$, but much closer to 0.5 and definitely not in the transition region. They argued that the optimal λ value for a task should strongly depend on the task. Specifically in the task considered by Packard one would expect a λ value close to 0.5 for a well performing rule, because the task is symmetric with respect to the exchange of ones and zeros. A rule with $\lambda < 0.5$ tends to decrease the number of 1's in the state vector because more entries in the rule table map the state to 0. This can lead to errors if the number of ones in the initial state is slightly larger than 0.5. Indeed, a rule which performs very well on this task, the Gacs-Kurdyumov-Levin (GKL) rule, has $\lambda = 0.5$. It was suggested that artefacts in the genetic algorithm could account for the different results.

We want to return here to the notion of computation. Wolfram and Langton were interested in universal computations. Although universality results for automata are mathematically interesting, they do not contribute much to the goal of understanding computations in biological neural system. Biological organisms usually face computational tasks which are quite different from the off-line computations on discrete batch inputs for which Turing machines are designed.

Packard was interested in automata which perform a specific kind of computation with the transition function being the "program". Mitchell at al. showed that there are complex tasks for which the best systems are not located at the edge of chaos. In (Mitchell et al., 1993), a third meaning of computation in cellular automata — a kind of "intrinsic" computation – is mentioned: "Here, computation is not

6. In the case of two-state cellular automata, high λ values imply that most state transitions map to the single non-quiescent state which leads to ordered dynamics. The most heterogeneous rules are found at $\lambda = 0.5$.

interpreted as the performance of a 'useful' transformation of the input to produce the output. Rather, it is measured in terms of generic, structural computational elements such as memory, information production, information transfer, logical operations, and so on. It is important to emphasize that the measurement of such intrinsic computational elements does not rely on a semantics of utility as do the preceding computational types." (Mitchell et al., 1993). It is worthwhile to note that this "intrinsic" computations in dynamical systems can be used by a readout unit which maps system states to desired outputs. This is the basic idea of the liquid state machine and echo state networks, and it is the basis of the considerations in the following sections.

To summarize, systems at the edge of chaos are believed to be computationally powerful. However, the type of computations considered so far are considerably different from computations in organisms. In the following section, we will consider a model of computation better suited for our purposes.

1.6 The Edge of Chaos in Systems with Online Input Streams

All previously considered computations were off-line computations where some initial state (the input) is transformed by the dynamics into a terminal state or state cycle (the output). However, computation in biological neural networks is quite different from computations in Turing machines or other traditional computational models. The input to an organism is a continuous stream of data and the organism reacts in real-time (i.e., within a given time interval) to information contained in this input. Hence, as opposed to batch processing, the input to a biological system is a time varying signal which is mapped to a time varying output signal. Such mappings are also called filters. In this section, we will have a look at recent work on real-time computations in threshold networks by Bertschinger and Natschläger (2004) (see also (Natschläger et al., 2005)). Results of experiments with closely related hardware models are reported in (Schuermann et al., 2005).

Threshold networks are special cases of Boolean networks consisting of N elements (units) with states $x_i \in \{-1, 1\}, i = 1, \ldots, N$. In networks with online input, the state of each element depends on the state of exactly K randomly chosen other units and in addition on an external input signal $u(\cdot)$ (the online input). At each time step, $u(t)$ assumes the value $\bar{u} + 1$ with probability r and the value $\bar{u} - 1$ with probability $1 - r$. Here, $\bar{u}$ is a constant input bias. The transfer function of the elements is not an arbitrary Boolean function but a randomly chosen threshold function of the form

$$x_i(t+1) = \Theta\left(\sum_{j=1}^{N} w_{ij} x_j(t) + u(t+1)\right) \tag{1.3}$$

where $w_{ij} \in \mathbb{R}$ is the weight of the connection from element j to element i and $\Theta(h) = +1$ if $h \geq 0$ and $\Theta(h) = -1$ otherwise. For each element, exactly K of its

incoming weights are nonzero and chosen from a Gaussian distribution with zero mean and variance σ^2. Different dynamical regimes of such circuits are shown in Figure 1.3. The top row shows the online input and below typical activity patterns of networks with ordered, critical, and chaotic dynamics. The system parameters for each of these circuits are indicated in the phase plot below. The variance σ^2 of nonzero weights was varied to achieve the different dynamics. The transition from the ordered to the chaotic regime is referred to as the *critical line*.

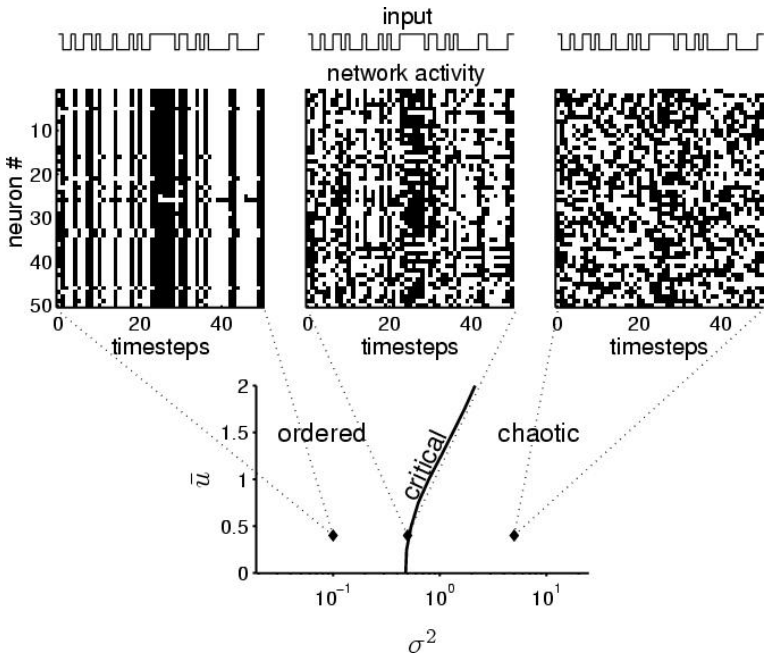

Figure 1.3 Threshold networks with online input streams in different dynamical regimes. The top row shows activity patterns for ordered (left), critical (middle), and chaotic behavior (right). Each vertical line represents the activity in one time step. Black (white) squares represent sites with value 1 (-1). Successive vertical lines represent successive circuit states. The input to the network is shown above the plots. The parameters σ^2 and $\bar{u}$ of these networks are indicated in the phase plot below. Further parameters: Number of input connections $K = 4$, number of elements $N = 250$.

Bertschinger and Natschläger used the approach of Derrida to determine the dynamical regime of these systems. They analyzed the change in Hamming distance between two (initial) states and their successor states provided that the same input is applied in both situations. Using Derridas annealed approximation, one can calculate the Hamming distance $d(t + 1)$ given the Hamming distance $d(t)$ of the states at time t. If arbitrarily small distances tend to increase, the network operates in the chaotic phase. If arbitrarily small distances tend to decrease, the

network operates in the ordered phase. This can also be expressed by the stability of the fixed point $d^* = 0$. In the ordered phase, this fixed point is the only fixed point and it is stable. In the chaotic phase, another fixed point appears and $d^* = 0$ becomes unstable. The fixed point 0 is stable if the absolute value of the slope of the map at $d(t) = 0$

$$\alpha = \frac{\partial d(t+1)}{\partial d(t)}\Big|_{d(t)=0}$$

is smaller than 1. Therefore, the transition from order to chaos (the critical line) is given by the line $|\alpha| = 1$. This line can be characterized by the equation

$$rP_{BF}(\bar{u}+1) + (1-r)P_{BF}(\bar{u}-1) = \frac{1}{K}, \tag{1.4}$$

where the bit-flip probability $P_{BF}(v)$ is the probability that a single changed state component in the K inputs to a unit that receives the current online input v leads to a change of the output of that unit. This result has a nice interpretation. Consider a value of $r = 1$, i.e. the input to the network is constant. Consider two network states $\mathcal{C}_1$, $\mathcal{C}_2$ which differ only in one state component. This different component is on average mapped to K elements (because each gate receives K inputs, hence there are altogether $N \cdot K$ connections). If the bit-flip probability in each of these units is larger than $\frac{1}{K}$, then more than one of these units will differ on average in the successors states $\mathcal{C}_1'$, $\mathcal{C}_2'$. Hence, differences are amplified. If the bit-flip probability of each element is smaller than $\frac{1}{K}$, the differences will die out on average.

1.7 Real-Time Computation in Dynamical Systems

In the previous section we were interested in the dynamical properties of systems with online input. The work we discussed there was influenced by recent ideas concerning computation in neural circuits that we will sketch in this section.

The idea to use the rich dynamics of neural systems which can be observed in cortical circuits rather than to restrict them resulted in the "liquid state machine" model by Maass et al. (2002) and the "echo state network" by Jäger (2002).[7] They assume time series as inputs and outputs of the system. A recurrent network is used to hold nonlinearly transformed information about the past input stream in the state of the network. It is followed by a memoryless readout unit which simply looks at the current state of the circuit. The readout can then learn to map the current state of the system onto some target output. Superior performance of echo state networks for various engineering applications is suggested by the results of

7. The model in (Maass et al., 2002) was introduced in the context of biologically inspired neural microcircuits. The network consisted of spiking neurons. In (Jäger, 2002), the network consisted of sigmoidal neurons.

Jäger and Haas (2004).

The requirement that the network is operating in the ordered phase is important in these models, although it is usually described with a different terminology. The ordered phase can be described by using the notion of *fading memory* (Boyd and Chua, 1985). Time invariant fading memory filters are exactly those filters which can be represented by Volterra series. Informally speaking, a network has fading memory it its state at time t depends (up to some finite precision) only on the values (up to some finite precision) of its input from some finite time window $[t-T, t]$ into the past (Maass et al., 2002). This is essentially equivalent to the requirement that if there are no longer any differences in the online inputs then the state differences converge to 0, which is called "echo state property" in (Jäger, 2002).

Besides the fading memory property, another property of the network is important for computations on time series: the pairwise separation property (Maass et al., 2002). Roughly speaking, a network has the pairwise separation property if for any two input time series which differed in the past, the network assumes at subsequent time points different states.

Chaotic networks have such separation property, but they do not have fading memory since differences in the initial state are amplified. On the other hand, very ordered systems have fading memory but provide weak separation. Hence, the separation property and the fading memory property are antagonistic. Ideally, one would like to have high separation on salient differences in the input stream but still keep the fading memory property (especially for variances in the input stream that do not contribute salient information). It is therefore of great interest to analyze these properties in models for neural circuits.

A first step in this direction was made in (Bertschinger and Natschläger, 2004) in the context of threshold circuits. Similar to Section 1.6, one can analyze the evolution of the state separation resulting from two input streams u_1 and u_2 which differ at time t with some probability. The authors defined the "network mediated separation" (short: NM-separation) of a network. Informally speaking, the NM-separation is roughly the amount of state distance in a network which results from differences in the input stream minus the amount of state difference resulting from different initial states. Hence, the NM-separation has a small value in the ordered regime, where both terms are small, but also in the chaotic regime, where both terms are large. Indeed, it was shown that the NM-separation peaks at the critical line, which is shown in Figure 1.4a. Hence, Bertschinger and Natschläger (2004) offer a new interpretation for the critical line and provide a more direct link between the edge of chaos and computational power.

Since the separation property is important for the computational properties of the network, one would expect that the computational performance peaks near the critical line. This was confirmed with simulations where the computational task was to compute the delayed 3-bit parity[8] of the input signal. The readout neuron

8. The delayed 3-bit parity of an input signal $u(\cdot)$ is given by $PARITY(u(t-\tau), u(t-$

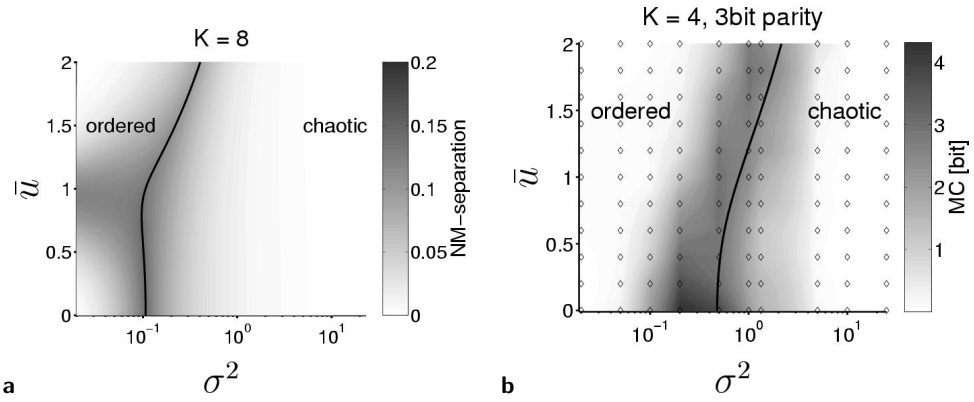

Figure 1.4 The network mediated separation and computational performance for a 3-bit parity task with different settings of parameters σ^2 and $\bar{u}$. **a)** The NM-separation peaks at the critical line. **b)** High performance is achieved near the critical line. The performance is measured in terms of the memory capacity MC (Jäger, 2002). The memory capacity is defined as the mutual information MI between the network output and the target function summed over all delays $\tau > 0$ on a test set. More formally, $MC = \sum_{\tau=0}^{\infty} MI(v_\tau, y_\tau)$, where $v_\tau(\cdot)$ denotes the network output and $y_\tau(t) = PARITY(u(t-\tau), u(t-\tau-1), u(t-\tau-2))$ is the target output.

was implemented by a simple linear classifier $C(\mathbf{x}(t)) = \Theta(\mathbf{w} \cdot \mathbf{x}(t) + w_0)$ which was trained with linear regression. Note that the parity task is quite complex since it partitions the set of all inputs into two classes which are not linearly separable (and can therefore not be represented by the linear readout alone), and it requires memory. Figure 1.4b shows that the highest performance is achieved for parameter values close to the critical line, although it is not clear why the performance drops for increasing values of $\bar{u}$. In contrast to preceding work (Langton, 1990; Packard, 1988), the networks used were not optimized for a specific task. Only the linear readout was trained to extract the specific information from the state of the system. This is important since it decouples the dynamics of the network from a specific task.

1.8 Self-Organized Criticality

Are there systems in nature with dynamics located at the edge of chaos? Since the edge of chaos is a small boundary region in the space of possible dynamics, only a vanishing small fraction of systems should operate in this dynamical regime. However, it was argued that such "critical" systems are abundant in nature,

$\tau - 1), u(t-\tau-2))$ for delays $\tau > 0$. The function $PARITY$ outputs 1 if the number of inputs which assume the value $\bar{u} + 1$ is odd and -1 otherwise.

see (Bak et al., 1988). How is this possible if critical dynamics may occur only accidentally in nature? Bak and collaborators argue that a class of dissipative coupled systems naturally evolve towards critical dynamics (Bak et al., 1988). This phenomenon was termed self-organized criticality (SOC) and it was demonstrated with a model of a sand pile. Imagine to build up a sand pile by randomly adding sand to the pile, a grain at a time. As sand is added, the slope will increase. Eventually, the slope will reach a critical value. Whenever the local slope of the pile is too steep, sand will slide off, therefore reducing the slope locally. On the other hand, if one starts with a very steep pile it will collapse and reach the critical slope from the other direction.

In neural systems, the topology of the network and the synaptic weights strongly influence the dynamics. Since the amount of genetically determined connections between neurons is limited, self-organizing processes during brain development as well as learning processes are assumed to play a key role in regulating the dynamics of biological neural networks (Bornholdt and Röhl, 2003). Although the dynamics is a global property of the network, biologically plausible learning rules try to estimate the global dynamics from information available at the local synaptic level and they only change local parameters. Several SOC rules have been suggested (Christensen et al., 1998; Bornholdt and Rohlf, 2000; Bornholdt and Röhl, 2003; Natschläger et al., 2005). In (Bornholdt and Röhl, 2003), the degree of connectivity was regulated in a locally connected network (i.e. only neighboring neurons are connected) with stochastic state update dynamics. A local rewiring rule was used which is related to Hebbian learning. The main idea of this rule is that the average correlation between the activities of two neurons contains information about the global dynamics. This rule only relies on information available on the local synaptic level.

Self-organized criticality in systems with online input streams (as discussed in Section 1.6) was considered in (Natschläger et al., 2005). According to Section 1.6, the dynamics of a threshold network is at the critical line if the bit-flip probability P_{BF} (averaged over the external and internal input statistics) is equal to $\frac{1}{K}$, where K is the number of inputs to a unit. The idea is to estimate the bit-flip probability of a unit by the mean distance of the internal activation of that unit from the firing threshold. This distance is called the *margin*. Intuitively, a node with an activation much higher or lower than its firing threshold is rather unlikely to change its output if a single bit in its inputs is flipped. Each node i then applies synaptic scaling to its weights w_{ij} in order to adjust itself towards the critical line:

$$w_{ij}(t+1) = \begin{cases} \frac{1}{1+\nu} \cdot w_{ij}(t) & \text{if } P_{BF}^{est_i}(t) > \frac{1}{K} \\ (1+\nu) \cdot w_{ij}(t) & \text{if } P_{BF}^{est_i}(t) < \frac{1}{K} \end{cases} \tag{1.5}$$

where $0 < \nu \ll 1$ is the learning rate and $P_{BF}^{est_i}(t)$ is an estimate of the bit-flip probability P_{BF}^{i} of unit i. It was shown by simulations that this rule keeps the dynamics in the critical regime, even if the input statistics changes. The computational capabilities of randomly chosen circuits with this synaptic scaling

rule acting online during computation was tested in a setup similar to that discussed in Section 1.7. The performance of these networks was as high as for circuits where the parameters were a priori chosen in the critical regime, and they stayed in this region. This shows that systems can perform specific computations while still being able to react to changing input statistics in a flexible way.

1.9 Towards the Analysis of Biological Neural Systems

Do cortical microcircuits operate at the edge of chaos? If biology makes extensive use of the rich internal dynamics of cortical circuits, then the previous considerations would suggest this idea. However, the neural elements in the brain are quite different from the elements discussed so far. Most importantly, biological neurons communicate with spikes, discrete events in continuous time. In this section, we will investigate the dynamics of spiking circuits and ask: In what dynamical regimes are neural microcircuits computationally powerful? We propose in this section a conceptual framework and new quantitative measures for the investigation of this question (see also Maass et al. (2005)).

In order to make this approach feasible, in spite of numerous unknowns regarding synaptic plasticity and the distribution of electrical and biochemical signals impinging on a cortical microcircuit, we make in the present first step of this approach the following simplifying assumptions:

1. Particular neurons ("readout neurons") learn via synaptic plasticity to extract specific information encoded in the spiking activity of neurons in the circuit.

2. We assume that the cortical microcircuit itself is highly recurrent, but that the impact of feedback that a readout neuron might send back into this circuit can be neglected.[9]

3. We assume that synaptic plasticity of readout neurons enables them to learn arbitrary linear transformations. More precisely, we assume that the input to such readout neuron can be approximated by a term $\sum_{i=1}^{n-1} w_i x_i(t)$, where $n-1$ is the number of presynaptic neurons, $x_i(t)$ results from the output spike train of the ith presynaptic neuron by filtering it according to the low-pass filtering property of the membrane of the readout neuron,[10] and w_i is the efficacy of the synaptic connection. Thus $w_i x_i(t)$ models the time course of the contribution of previous spikes from the ith presynaptic neuron to the membrane potential at the soma of

9. This assumption is best justified if such readout neuron is located for example in another brain area that receives massive input from many neurons in this microcircuit and only has diffuse backwards projection. But it is certainly problematic and should be addressed in future elaborations of the present approach.

10. One can be even more realistic and filter it also by a model for the short term dynamics of the synapse into the readout neuron, but this turns out to make no difference for the analysis proposed in this chapter.

this readout neuron. We will refer to the vector $\mathbf{x}(t)$ as the *circuit state at time t* (although it is really only that part of the circuit state which is directly observable by readout neurons).

All microcircuit models that we consider are based on biological data for generic cortical microcircuits (as described in Section 1.9.1), but have different settings of their parameters.

1.9.1 Models for generic cortical microcircuits

Our empirical studies were performed on a large variety of models for generic cortical microcircuits (we refer to (Maass et al., 2004) for more detailed definitions and explanations). All circuit models consisted of leaky-integrate-and-fire neurons [11] and biologically quite realistic models for dynamic synapses.[12] Neurons (20 % of which were randomly chosen to be inhibitory) were located on the grid points of a 3D grid of dimensions $6 \times 6 \times 15$ with edges of unit length. The probability of a synaptic connection from neuron a to neuron b was proportional to $exp(-D^2(a,b)/\lambda^2)$, where $D(a,b)$ is the Euclidean distance between a and b, and λ is a spatial connectivity constant (not to be confused with the λ-parameter used by Langton). Synaptic efficiencies w were chosen randomly from distributions that reflect biological data (as in (Maass et al., 2002)), with a common scaling factor W_{scale}.

Linear readouts from circuits with $n-1$ neurons were assumed to compute a weighted sum $\sum_{i=1}^{n-1} w_i x_i(t) + w_0$ (see Section 1.9). In order to simplify notation we assume that the vector $\mathbf{x}(t)$ contains an additional constant component $x_0(t) = 1$, so that one can write $\mathbf{w} \cdot \mathbf{x}(t)$ instead of $\sum_{i=1}^{n-1} w_i x_i(t) + w_0$. In the case of classification tasks we assume that the readout outputs 1 if $\mathbf{w} \cdot \mathbf{x}(t) \geq 0$, and 0 otherwise.

In order to investigate the influence of synaptic connectivity on computational performance, neural microcircuits were drawn from this distribution for 10 different values of λ (which scales the number and average distance of synaptically connected neurons) and 9 different values of W_{scale} (which scales the efficacy of all synaptic connections). 20 microcircuit models C were drawn for each of these 90 different assignments of values to λ and W_{scale}. For each circuit a linear readout was trained to perform one (randomly chosen) out of 2^{80} possible classification tasks on noisy variations u of 80 fixed spike patterns as circuit inputs u. See Figure 1.5 for two examples of such spike patterns. The target performance of any such circuit was

11. Membrane voltage V_m modeled by $\tau_m \frac{dV_m}{dt} = -(V_m - V_{resting}) + R_m \cdot (I_{syn}(t) + I_{background} + I_{noise})$, where $\tau_m = 30$ ms is the membrane time constant, I_{syn} models synaptic inputs from other neurons in the circuits, $I_{background}$ models a constant unspecific background input and I_{noise} models noise in the input. The membrane resistance R_m was chosen as $1M\Omega$.

12. Short term synaptic dynamics was modeled according to (Markram et al., 1998), with distributions of synaptic parameters U (initial release probability), D (time constant for depression), F (time constant for facilitation) chosen to reflect empirical data (see Maass et al. (2002) for details).

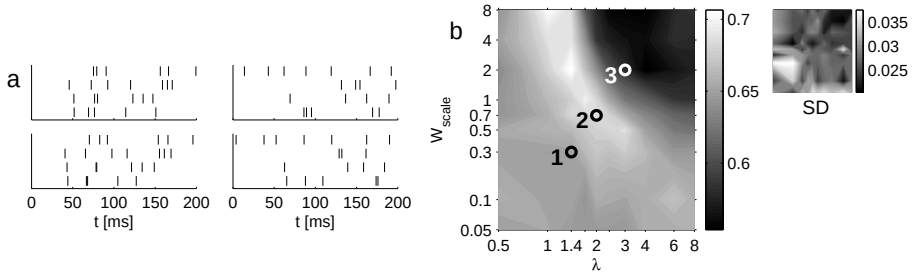

Figure 1.5 Performance of different types of neural microcircuit models for classification of spike patterns. **a)** In the top row are two examples of the 80 spike patterns that were used (each consisting of 4 Poisson spike trains at 20 Hz over 200 ms), and in the bottom row are examples of noisy variations (Gaussian jitter with SD 10 ms) of these spike patterns which were used as circuit inputs. **b)** Fraction of examples (for 200 test examples) that were correctly classified by a linear readout (trained by linear regression with 500 training examples). Results are shown for 90 different types of neural microcircuits C with λ varying on the x-axis and W_{scale} on the y-axis (20 randomly drawn circuits and 20 target classification functions randomly drawn from the set of 2^{80} possible classification functions were tested for each of the 90 different circuit types, and resulting correctness-rates were averaged). Circles mark three specific choices of λ, W_{scale}-pairs for comparison with other figures, see Figure 1.6. The standard deviation of the result is shown in the inset on the upper right.

to output at time $t = 200$ ms the class (0 or 1) of the spike pattern from which the preceding circuit input had been generated (for some arbitrary partition of the 80 fixed spike patterns into two classes). Each spike pattern u consisted of 4 Poisson spike trains over 200 ms. Performance results are shown in Figure 1.5b for 90 different types of neural microcircuit models.

1.9.2 Locating the edge of chaos in neural microcircuit models

It turns out that the previously considered characterizations of the edge of chaos are not too successful in identifying those parameter values in the map of Fig. 1.5b that yield circuits with large computational power (Maass et al., 2005). The reason is that large initial state differences (as they are typically caused by different spike input patterns) tend to yield for most values of the circuit parameters nonzero state differences not only while the online spike inputs are different, but also long afterwards when the online inputs agree during subsequent seconds (even if the random internal noise is identical in both trials). But if one applies the definition of the edge of chaos via Lyapunov exponents (see (Kantz and Schreiber, 1997)), the resulting edge of chaos lies for the previously introduced type of computations (classification of noisy spike templates by a trained linear readout) in the region of the best computational performance (see the map in Fig. 1.5b, which is repeated

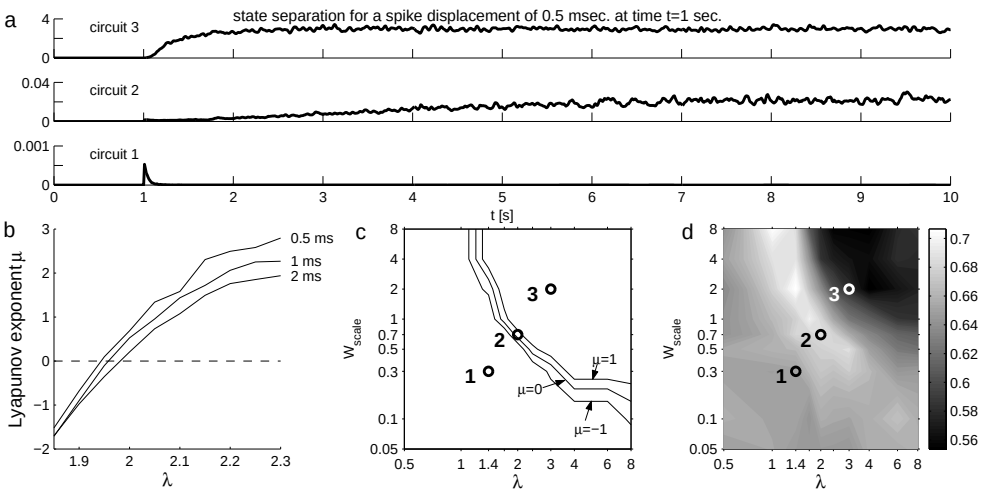

Figure 1.6 Analysis of small input differences for different types of neural microcircuit models as specified in Section 1.9.1. Each circuit C was tested for two arrays u and v of 4 input spike trains at 20 Hz over 10 s that differed only in the timing of a single spike at time $t = 1$ s. **a)** A spike at time $t = 1$ s was delayed by 0.5 ms. Temporal evolution of Euclidean differences between resulting circuit states $\mathbf{x}_u(t)$ and $\mathbf{x}_v(t)$ with 3 different values of λ, W_{scale} according to the 3 points marked in panel c). For each parameter pair, the average state difference of 40 randomly drawn circuits is plotted. **b)** Lyapunov exponents μ along a straight line between the points marked in panel c) with different delays of the delayed spike. The delay is denoted on the right of each line. The exponents were determined for the average state difference of 40 randomly drawn circuits. **c)** Lyapunov exponents μ for 90 different types of neural microcircuits C with λ varying on the x-axis and W_{scale} on the y-axis (the exponents were determined for the average state difference of 20 randomly drawn circuits for each parameter pair). A spike in u at time $t = 1$ s was delayed by 0.5 ms. The contour lines indicate where μ crosses the values -1, 0, and 1. **d)** Computational performance of these circuits (same as Figure 1.5b), shown for comparison with panel c).

for easier comparison in Fig. 1.6d). For this definition one looks for the exponent $\mu \in \mathbb{R}$ which provides through the formula

$$\delta_{\Delta T} \approx \delta_0 \cdot e^{\mu \Delta T}$$

the best estimate of the state separation $\delta_{\Delta T}$ at time ΔT after the computation was started in two trials with an initial state difference δ_0. We generalize this analysis to the case with online input by choosing exactly the same online input (and the same random noise) during the intervening time interval of length ΔT, and by averaging the resulting state differences $\delta_{\Delta T}$ over many random choices of such online inputs (and internal noise). As in the classical case with offline input it turns out to be essential to apply this estimate for $\delta_0 \to 0$, since $\delta_{\Delta T}$ tends to saturate for each

fixed value δ_0. This can be seen in Fig. 1.6a, which shows results of this experiment for a δ_0 that results from moving a single spike that occurs in the online input at time $t = 1$s by 0.5 ms. This experiment was repeated for 3 different circuits with parameters chosen from the 3 locations marked on the map in Fig. 1.6c. By determining the best fitting μ for $\Delta T = 1.5$s for 3 different values of δ_0 (resulting from moving a spike at time $t = 1$s by 0.5, 1, 2 ms) one gets the dependence of this Lyapunov exponent on the circuit parameter λ shown in Fig. 1.6b (for values of λ and W_{scale} on a straight line between the points marked in the map of Fig. 1.6c). The middle curve in Fig. 1.6c shows for which values of λ and W_{scale} the Lyapunov exponent is estimated to have the value 0. By comparing it with those regions on this parameter map where the circuits have the largest computational power (for the classification of noisy spike patterns, see Fig. 1.6d), one sees that this line runs through those regions which yield the largest computational power for these computations. We refer to (Mayor and Gerstner, 2005) for other recent work on studies of the relationship between the edge of chaos and the computational power of spiking neural circuit models.

Although this estimated edge of chaos coincides quite well with points of best computational performance, it remains an unsatisfactory tool for predicting parameter regions with large computational power for three reasons:

i) Since the edge of chaos is a lower dimensional manifold in a parameter map (in this case a curve in a 2D map), it cannot predict the (full dimensional) regions of a parameter map with high computational performance (e.g. the regions with light shading in Fig. 1.5b).

ii) The edge of chaos does not provide intrinsic reasons *why* points of the parameter map yield small or large computational power.

iii) It turns out that in some parameter maps *different* regions provide circuits with large computational power for *different* classes of computational tasks (as shown in (Maass et al., 2005) for computations on spike patterns and for computations with firing rates). But the edge of chaos can at best single out peaks for *one* of these regions. Hence it cannot possibly be used as a universal predictor of maximal computational power for all types of computational tasks.

These three deficiencies suggest that one has to think about different strategies to approach the central question of this chapter. The strategy we will pursue in the following is based on the assumption that the computational function of cortical microcircuits is not fully genetically encoded, but rather emerges through various forms of plasticity ("learning") in response to the actual distribution of signals that the neural microcircuit receives from its environment. From this perspective the question about the computational function of cortical microcircuits C turns into the questions:

a) What functions (i.e. maps from circuit inputs to circuit outputs) can the circuit C *learn* to compute.

b) How well can the circuit C generalize a specific learned computational function

to new inputs?

In the following, we propose quantitative criteria based on rigorous mathematical principles for evaluating a neural microcircuit C with regard to questions a) and b). We will compare in Section 1.9.5 the predictions of these quantitative measures with the actual computational performance achieved by neural microcircuit models as discussed in Section 1.9.1.

1.9.3 A measure for the kernel-quality

One expects from a powerful computational system that significantly different input streams cause significantly different internal states, and hence may lead to different outputs. Most real-world computational tasks require that the circuit gives a desired output not just for 2, but for a fairly large number m of significantly different inputs. One could of course test whether a circuit C can separate each of the $\binom{m}{2}$ pairs of such inputs. But even if the circuit can do this, we do not know whether a neural readout from such circuit would be able to produce given target outputs for these m inputs.

Therefore we propose here the *linear separation property* as a more suitable quantitative measure for evaluating the computational power of a neural microcircuit (or more precisely: the kernel-quality of a circuit; see below). To evaluate the linear separation property of a circuit C for m different inputs $u_1, \ldots, u_m$ (which are in the following always functions of time, i.e. input streams such as for example multiple spike trains) we compute the rank of the $n \times m$ matrix M whose columns are the circuit states $\mathbf{x}_{u_i}(t_0)$ resulting at some fixed time t_0 for the preceding input stream u_i. If this matrix has rank m, then it is *guaranteed* that *any* given assignment of target outputs $y_i \in \mathbb{R}$ at time t_0 for the inputs u_i can be implemented by this circuit C (in combination with a linear readout). In particular, each of the 2^m possible binary classifications of these m inputs can then be carried out by a *linear* readout from this fixed circuit C. Obviously such insight is much more informative than a demonstration that some *particular* classification task can be carried out by such circuit C. If the rank of this matrix M has a value $r < m$, then this value r can still be viewed as a measure for the computational power of this circuit C, since r is the number of "degrees of freedom" that a linear readout has in assigning target outputs y_i to these inputs u_i (in a way which can be made mathematically precise with concepts of linear algebra). Note that this rank-measure for the linear separation property of a circuit C may be viewed as an empirical measure for its *kernel-quality*, i.e. for the complexity and diversity of nonlinear operations carried out by C on its input stream in order to boost the classification power of a subsequent *linear* decision-hyperplane (see Vapnik, 1998).

1.9.4 A measure for the generalization-capability

Obviously the preceding measure addresses only one component of the computational performance of a neural circuit C. Another component is its capability to *generalize* a learnt computational function to *new* inputs. Mathematical criteria for generalization capability are derived in (Vapnik, 1998) (see Ch. 4 in Cherkassky and Mulier, 1998, for a compact account of results relevant for our arguments). According to this mathematical theory one can quantify the generalization capability of any learning device in terms of the VC-dimension of the class $\mathcal{H}$ of hypotheses that are potentially used by that learning device.[13] More precisely: if VC-dimension ($\mathcal{H}$) is substantially smaller than the size of the training set S_{train}, one can prove that this learning device generalizes well, in the sense that the hypothesis (or input-output map) produced by this learning device is likely to have for new examples an error rate which is not much higher than its error rate on S_{train}, provided that the new examples are drawn from the same distribution as the training examples (see Eqn. 4.22 in Cherkassky and Mulier, 1998).

We apply this mathematical framework to the class $\mathcal{H}_C$ of all maps from a set S_{univ} of inputs u (into $\{0,1\}$ which can be implemented by a circuit C. More precisely: $\mathcal{H}_C$ consists of all maps from S_{univ} into $\{0,1\}$ that a linear readout from circuit C with fixed internal parameters (weights etc.) but arbitrary weights $\mathbf{w} \in \mathbb{R}^n$ of the readout (that classifies the circuit input u as belonging to class 1 if $\mathbf{w} \cdot \mathbf{x}_u(t_0) \geq 0$, and to class 0 if $\mathbf{w} \cdot \mathbf{x}_u(t_0) < 0$) could possibly implement.

Whereas it is very difficult to achieve tight theoretical bounds for the VC-dimension of even much simpler neural circuits, see (Bartlett and Maass, 2003), one can efficiently estimate the VC-dimension of the class $\mathcal{H}_C$ that arises in our context for some finite ensemble S_{univ} of inputs (that contains all examples used for training or testing) by using the following mathematical result (which can be proved with the help of Radon's Theorem):

Theorem 1.1
Let r be the rank of the $n \times s$ matrix consisting of the s vectors $\mathbf{x}_u(t_0)$ for all inputs u in S_{univ} (we assume that S_{univ} is finite and contains s inputs). Then $r \leq$ VC-dimension$(\mathcal{H}_C) \leq r + 1$.

Proof idea. Fix some inputs $u_1, \ldots, u_r$ in S_{univ} so that the resulting r circuit states $\mathbf{x}_{u_i}(t_0)$ are linearly independent. The first inequality is obvious since this set of r linearly independent vectors can be shattered by linear readouts from the circuit C. To prove the second inequality one assumes for a contradiction that there exists a set $v_1, \ldots, v_{r+2}$ of $r + 2$ inputs in S_{univ} so that the corresponding

13. The VC-dimension (of a class $\mathcal{H}$ of maps H from some universe S_{univ} of inputs into $\{0,1\}$) is defined as the size of the largest subset $S \subseteq S_{univ}$ which can be *shattered* by $\mathcal{H}$. One says that $S \subseteq S_{univ}$ is shattered by $\mathcal{H}$ if for *every* map $f : S \to \{0,1\}$ there exists a map H in $\mathcal{H}$ such that $H(u) = f(u)$ for all $u \in S$ (this means that *every* possible binary classification of the inputs $u \in S$ can be carried out by some hypothesis H in $\mathcal{H}$).

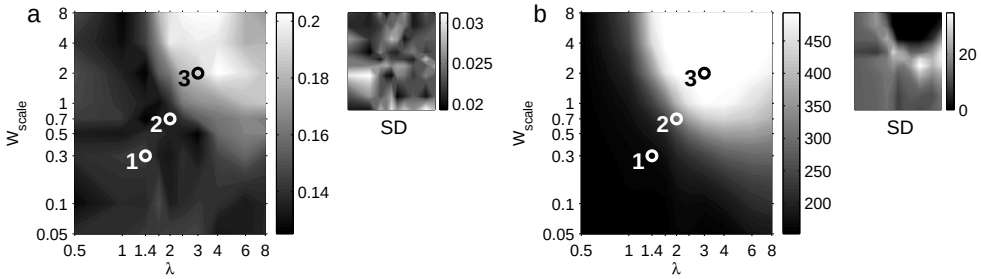

Figure 1.7 Measuring the generalization capability of neural microcircuit models. **a)** Test error minus train error (error was measured as the fraction of examples that were misclassified) in the spike pattern classification task discussed in Section 1.9.1 for 90 different types of neural microcircuits (as in Figure 1.5b). The standard deviation is shown in the inset on the upper right. **b)** Generalization capability for spike patterns: estimated VC-dimension of $\mathcal{H}_C$ (for a set S_{univ} of inputs u consisting of 500 jittered versions of 4 spike patterns), for 90 different circuit types (average over 20 circuits; for each circuit, the average over 5 different sets of spike patterns was used). The standard deviation is shown in the inset on the upper right. See Section 1.9.5 for details.

set of $r+2$ circuit states $\mathbf{x}_{v_i}(t_0)$ can be shattered by linear readouts. This set M of $r+2$ vectors is contained in the $r-$dimensional space spanned by the linearly independent vectors $\mathbf{x}_{u_1}(t_0), \ldots, \mathbf{x}_{u_r}(t_0)$. Therefore Radon's Theorem implies that M can be partitioned into disjoint subsets M_1, M_2 whose convex hulls intersect. Since these sets M_1, M_2 cannot be separated by a hyperplane, it is clear that no linear readout exists that assigns value 1 to points in M_1 and value 0 to points in M_2. Hence $M = M_1 \cup M_2$ is not shattered by linear readouts, a contradiction to our assumption. ■

We propose to use the rank r defined in Theorem 1.1 as an estimate of VC-dimension($\mathcal{H}_C$), and hence as a measure that informs us about the generalization capability of a neural microcircuit C. It is assumed here that the set S_{univ} contains many noisy variations of the same input signal, since otherwise learning with a randomly drawn training set $S_{train} \subseteq S_{univ}$ has no chance to generalize to new noisy variations. Note that each family of computational tasks induces a particular notion of what aspects of the input are viewed as noise, and what input features are viewed as signals that carry information which is relevant for the target output for at least one of these computational tasks. For example for computations on spike patterns some small jitter in the spike timing is viewed as noise. For computations on firing rates even the sequence of interspike intervals and temporal relations between spikes that arrive from different input sources are viewed as noise, as long as these input spike trains represent the same firing rates.

An example for the former computational task was discussed in Section 1.9.1. This task was to output at time $t = 200$ ms the class (0 or 1) of the spike pattern from which the preceding circuit input had been generated (for some arbitrary

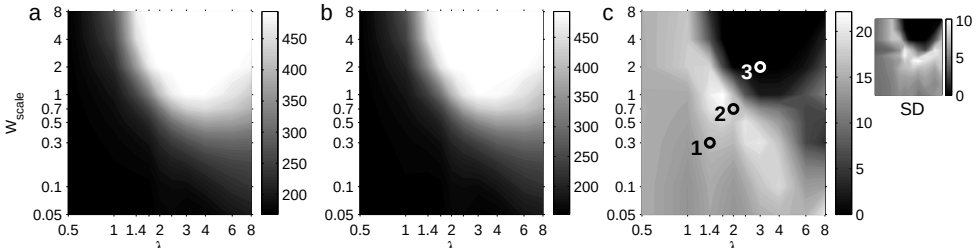

Figure 1.8 Values of the proposed measures for computations on spike patterns. **a)** Kernel-quality for spike patterns of 90 different circuit types (average over 20 circuits, mean $SD = 13$). **b)** Generalization capability for spike patterns: estimated VC-dimension of $\mathcal{H}_C$ (for a set S_{univ} of inputs u consisting of 500 jittered versions of 4 spike patterns), for 90 different circuit types (same as Figure 1.7b). **c)** Difference of both measures (the standard deviation is shown in the inset on the upper right). This should be compared with actual computational performance plotted in Figure 1.5b.

partition of the 80 fixed spike patterns into two classes, see Section 1.9.1). For a poorly generalizing network, the difference between train and test error is large. One would suppose that this difference becomes large as the network dynamics become more and more chaotic. This is indeed the case, see Figure 1.7a. The transition is is pretty well predicted by the estimated VC-dimension of $\mathcal{H}_C$, see Figure 1.7b.

1.9.5 Evaluating the influence of synaptic connectivity on computational performance

We now test the predictive quality of the two proposed measures for the computational power of a microcircuit on spike patterns. One should keep in mind that the proposed measures do not attempt to test the computational capability of a circuit for one particular computational task, but for *any* distribution on S_{univ} and for a very large (in general infinitely large) family of computational tasks that only have in common a particular bias regarding which aspects of the incoming spike trains may carry information that is relevant for the target output of computations, and which aspects should be viewed as noise. Figure 1.8a explains *why* the lower left part of the parameter map in Figure 1.5b is less suitable for any such computation, since there the kernel-quality of the circuits is too low.[14] Figure 1.8b explains *why* the upper right part of the parameter map in Figure 1.5b is less suitable, since a higher VC-dimension (for a training set of fixed size) entails poorer generalization capability. We are not aware of a theoretically founded way of combining both measures into a single value that predicts overall computational performance. But if one

14. The rank of the matrix consisting of 500 circuit states $\mathbf{x}_u(t)$ for $t = 200$ ms was computed for 500 spike patterns over 200 ms as described in Section 1.9.3, see Figure 1.5a. For each circuit, the average over 5 different sets of spike patterns was used.

just takes the difference of both measures (after scaling each linearly into a common range [0,1]) then the resulting number (see Figure 1.8c) predicts quite well which types of neural microcircuit models perform well for the particular computational tasks considered in Figure 1.5b.[15]

Results of further tests of the predictive power of these measures are reported in (Maass et al., 2005). These tests have been applied there to a completely different parameter map, and diverse classes of computational tasks.

1.10 Conclusions

The need to understand computational properties of complex dynamical systems is becoming more urgent. New experimental methods provide substantial insight into the inherent dynamics of the computationally most powerful classes of dynamical systems that are known: neural systems and gene regulation networks of biological organisms. More recent experimental data show that simplistic models for computations in such systems are not adequate, and that new concepts and methods have to be developed in order to understand their computational function. This short review has shown that several old ideas regarding computations in dynamical systems receive new relevance in this context, once they are transposed into a more realistic conceptual framework that allows us to analyze also online computations on continuous input streams. Another new ingredient is the investigation of the temporal evolution of information in a dynamical system from the perspective of models for the (biological) user of such information, i.e. from the perspective of neurons that receive inputs from several thousand presynaptic neurons in a neural circuit, and from the perspective of gene regulation mechanisms that involve thousands of transcription factors. Empirical evidence from the area of machine learning supports the hypothesis that readouts of this type, which are able to sample not just 2 or 3, but thousands of coordinates of the state vector of a dynamical system, impose different (and in general less obvious) constraints on the dynamics of a high dimensional dynamical system in order to employ such system for complex computations on continuous input streams. One might conjecture that unsupervised learning and regulation processes in neural systems adapt the system dynamics in such a way that these constraints are met. Hence suitable variations of the idea of self organized criticality may help us to gain a system-level perspective of synaptic plasticity and other adaptive processes in neural systems.

15. Similar results arise if one records the analog values of the circuit states with a limited precision of say 1%.

References

S. Amari. Characteristics of random nets of analog neuron-like elements. *IEEE Tranactions on Systems, Man, and Cybernetics*, 2:643–657, 1972.

P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality. *Physical Review A*, 38(1):364–378, 1988.

P. L. Bartlett and W. Maass. Vapnik-Chervonenkis dimension of neural nets. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 1188–1192. MIT Press (Cambridge), 2nd edition, 2003.

N. Bertschinger and T. Natschläger. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7):1413–1436, 2004.

S. Bornholdt and T. Röhl. Self-organized critical neural networks. *Physical Review E*, 67:066118, 2003.

S. Bornholdt and T. Rohlf. Topological evolution of dynamical networks: Global criticality from local dynamics. *Physical Review Letters*, 84(26):6114–6117, 2000.

S. Boyd and L. O. Chua. Fading memory and the problem of approximating nonlinear oparators with Volterra series. *IEEE Trans. on Circuits and Systems*, 32:1150–1161, 1985.

D. V. Buonomano and M. M. Merzenich. Temporal information transformed into a spatial code by a neural network with realistic properties. *Science*, 267:1028–1030, 1995.

V. Cherkassky and F. Mulier. *Learning from Data*. Wiley, New York, 1998.

K. Christensen, R. Donangelo, B. Koiller, and K. Sneppen. Evolution of random networks. *Physical Review Letters*, 81:2380, 1998.

E. F. Codd. *Cellular Automata*. Academic Press, New York, 1968.

Cowan J. D. Statistical mechanics of nervous nets. In E. R. Caianiello, editor, *Neural Networks*, pages 181–188. Springer-Verlag (Berlin), 1968.

B. Derrida and Y. Pomeau. Random networks of automata: A simple annealed approximation. *Europhysics Letters*, 1(2):45–49, 1986.

W. J. Freeman. *Mass Action in the Nervous System*. Academic Press (New York), 1975.

W. J. Freeman. Mescoscopic neurodynamics: From neuron to brain. *J. Physiology*, 94:303–320, 2000.

S. Grossberg. Nonlinear difference-differential equations in prediction and learning theory. *Proc. Nat. Acad. Sci. USA*, 58:1329–1334, 1967.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci. USA*, 79:2554–2558, 1982.

J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Nat. Acad. Sci. USA*, 81:3088–3092, 1984.

J. J. Hopfield and D. W. Tank. "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

J. J. Hopfield and D. W. Tank. Computing with neural circuits: A model. *Science*, 233:625–633, 1986.

H. Jäger. Short term memory in echo state networks. GMD Report 152, German National Research Center for Information Technology, 2002.

H. Jäger and H. Haas. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78–80, 2004.

P. Joshi and W. Maass. Movement generation with circuits of spiking neurons. *Neural Computation*, 2004. in press.

H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, Cambridge, 1997.

D. Kaplan and L. Glass. *Understanding Nonlinear Dynamics*. Springer, 1995.

S. A. Kauffman. Metabolic stability and epigenesis in randomly connected nets. *J. Theoret. Biol.*, 22:437, 1969.

S. A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.

C. G. Langton. Computation at the edge of chaos. *Physica D*, 42:12–37, 1990.

R. A. Legenstein, H. Markram, and W. Maass. Input prediction and autonomous movement analysis in recurrent circuits of spiking neurons. *Reviews in the Neurosciences (Special Issue on Neuroinformatics of Neural and Artificial Computation)*, 14(1–2):5–19, 2003.

W.A. Little. The existence of persistent states in the brain. *Mathematical Biosciences*, 19:101–120, 1974.

W. Maass, R. A. Legenstein, and N. Bertschinger. Methods for estimating the computational power and generalization capability of neural microcircuits. In *Proc. of NIPS 2004, Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005. URL `http://www.igi.tugraz.at/maass/psfiles/160_col.pdf`.

W. Maass and H. Markram. Theory of the computational function of microcircuit dynamics. In *Proc. of the 2004 Dahlem Workshop on Microcircuits*. MIT Press, 2005. URL `http://www.igi.tugraz.at/maass/psfiles/157_v2_web.pdf`.

W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural*

Computation, 14(11):2531–2560, 2002.

W. Maass, T. Natschläger, and H. Markram. Computational models for generic cortical microcircuits. In J. Feng, editor, *Computational Neuroscience: A Comprehensive Approach*, chapter 18, pages 575–605. Chapman & Hall/CRC, 2004.

H. Markram, Y. Wang, and M. Tsodyks. Differential signaling via the same axon of neocortical pyramidal neurons. *PNAS*, 95:5323–5328, 1998.

J. Mayor and W. Gerstner. Signal buffering in random networks of spiking neurons: microscopic vs. macroscopic phenomena. Preprint EPFL Lausanne, 2005.

M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7:89–130, 1993.

T. Natschläger, N. Bertschinger, and R. Legenstein. At the edge of chaos: Real-time computations and self-organized criticality in recurrent neural networks. In *Proc. of NIPS 2004, Advances in Neural Information Processing Systems*. MIT Press, 2005. to appear.

N. Packard. Adaption towards the edge of chaos. In J. A. S. Kelso, A. J. Mandell, and M. F. Shlesinger, editors, *Dynamic Patterns in Complex Systems*, pages 293–301. World Scientific, 1988.

F. Schuermann, K. Meier, and J. Schemmel. Edge of chaos computation in mixed-mode VLSI: a hard liquid. In *Proc. of NIPS 2004, Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005. to appear.

A. R. Smith. Simple computation-universal cellular spaces. *Journal of the ACM*, 18(3):339–353, 1971.

S. H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications in Physics, Biology, Chemistry, and Engineering (Studies in Nonlinearity)*. Addison-Wesley, 1994.

V. N. Vapnik. *Statistical Learning Theory*. John Wiley (New York), 1998.

J von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, Illinois, 1966.

S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.