nature neuroscience

Building functional networks of spiking model neurons

L F Abbott[1,2], Brian DePasquale[1] & Raoul-Martin Memmesheimer[1,3]

Most of the networks used by computer scientists and many of those studied by modelers in neuroscience represent unit activities as continuous variables. Neurons, however, communicate primarily through discontinuous spiking. We review methods for transferring our ability to construct interesting networks that perform relevant tasks from the artificial continuous domain to more realistic spiking network models. These methods raise a number of issues that warrant further theoretical and experimental study.

The world around us is described by continuous variables—distances, angles, wavelengths, frequencies—and we respond to it with continuous motions of our bodies. Yet the neurons that represent and process sensory information and generate motor acts communicate with each other almost exclusively through discrete action potentials. The use of spikes to represent, process and interpret continuous quantities and to generate smooth and precise motor acts is a challenge both for the nervous system and for those who study it. A related issue is the wide divergence between the timescales of action potentials and of perceptions and actions. How do millisecond spikes support the integration of information and production of responses over much longer times? Theoretical neuroscientists address these issues by studying networks of spiking model neurons. Before this can be done, however, network models with functionality over behaviorally relevant timescales must be constructed. Here, we review a number of methods that have been developed for building recurrent network models of spiking neurons.

Constructing a network requires choosing the models used to describe its individual neurons and synapses, defining its pattern of connectivity, and setting its many parameters (**Fig. 1a**). The networks we discuss are based on model neurons and synapses that are, essentially, as simple as possible. The complexity of these networks resides in the patterns and strengths of the connections between neurons (although we consider dendritic processing toward the end of this Perspective article). This should not be interpreted as a denial of the importance or the complexity of the dynamics of membrane and synaptic conductances, or of phenomena such as bursting, spike-rate adaptation, neuromodulation and synaptic plasticity. These are undoubtedly important, but the simplified models we discuss allow us to assess how much of the dynamics needed to support temporally extended behaviors can be explained by network connectivity. Furthermore, such models provide a foundation upon which more complex descriptions can be developed.

The problem we are addressing is this: a network receives an input $f_{in}(t)$, and its task is to generate a specified output $f_{out}(t)$ (**Fig. 1a**; we discuss below how this output is computed). Our job is to configure the network so that it does this task, where by 'configure' we mean set the weights (that is, strengths) of the network synapses to appropriate values. For a network of N neurons, these weights are given by the elements of an $N \times N$ matrix, denoted by J, that describes the modifiable connections between network neurons (although some of these elements may be constrained to 0, corresponding to nonexistent connections). We note here that we are constructing recurrently connected networks, which pose unique challenges not faced when constructing feedforward networks. Given our interest in spanning the temporal gap between spikes and behavior, tasks of interest often involve integrating an input over time[1–12], responding to particular temporal input sequences[13–15], responding after a delay or with an activity sequence[13,16–24], responding with a temporally complex output[7,22–29] or autonomously generating complex dynamics[6,10,22–24]. In this Perspective, we focus on general approaches that extend our ability to construct spiking networks capable of performing a wide variety of tasks or that make spiking networks perform these tasks more accurately.

Determining the connection matrix required to make a network perform a particular task is difficult because it is not obvious what the individual neurons of the network should do to generate the desired output while supporting each others' activities. This is the classic credit-assignment problem of network learning: what should each individual neuron do to contribute to the collective cause of performing the task? The field of machine learning has addressed credit assignment by developing error gradient–based methods, such as back-propagation, that have been applied with considerable success[30]. This approach has also been used to construct abstract network models known as rate models in which neurons communicate with each other through continuous variables[31,32]. Unfortunately, the application of gradient-based methods to spiking networks[28,33–35] is problematic because it has not been clear how to define an appropriate differentiable error measure for spike trains. The methods that we review here can all be thought of as ways to solve the credit assignment problem without resorting to gradient-based procedures.

Defining the input, output and network connections

Before beginning the general discussion, we need to explain how neurons in the network interact, how they receive an input and how they produce an output. This, in turn, requires us to define what we call the normalized synaptic current, $s(t)$, that arises from a spike train (**Fig. 2a**, top and middle traces). In the synapse model we use, each

[1]Department of Neuroscience, Columbia University College of Physicians and Surgeons, New York, New York, USA. [2]Department of Physiology and Cellular Biophysics, Columbia University College of Physicians and Surgeons, New York, New York, USA. [3]Department for Neuroinformatics, Donders Institute for Brain Cognition and Behavior, Radboud University, Nijmegen, the Netherlands. Correspondence should be addressed to L.F.A. (lfa2103@columbia.edu).

Received 25 November 2015; accepted 11 January 2016; published online 23 February 2016; doi:10.1038/nn.4241

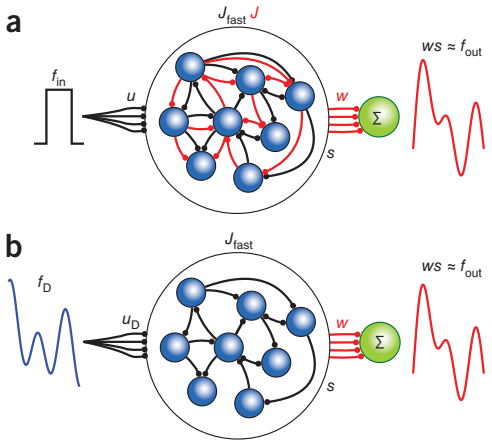

Figure 1 Structure of autonomous and driven networks. (**a**) The autonomous network. In this diagram, black lines and dots denote fixed connections, and red lines and dots are connections that are adjusted to make the network function properly. A defined input f_{in} is provided to the network through connections characterized by weights u. Neurons in the network are connected by two types of synapses, parameterized by J_{fast} (in black) and J (in red). The problem is to choose the strengths of the synapses defined by J, and the weights w, so that the output of the network, ws, approximates a given target output f_{out}. (**b**) The driven network. In this case, the network is driven by input f_D, through weights u_D, that forces it to produce the desired output. Only the fixed synapses denoted by J_{fast} are included. Output weights are adjusted as in the autonomous network.

presynaptic spike causes the normalized synaptic current to increase instantaneously by 1, $s \rightarrow s + 1$. Between spikes, s decays exponentially toward 0 with a time constant τ, which is set to 100 ms in the examples we show. There is one normalized synaptic current for each network neuron, so s is an N-component vector. The normalized synaptic current is used to construct both the output of the network and the inputs that each neuron receives through the synapses described by the matrix J (**Fig. 1a**). The synaptic current generated in a postsynaptic neuron by a particular presynaptic neuron is given by the appropriate synaptic weight multiplied by the normalized synaptic current for that presynaptic neuron. The synaptic currents for all the network neurons are given collectively by Js.

All of the models we present have, in addition to the connections described by J, a second set of synapses with time constants considerably faster than τ, described by J_{fast}. We consider two arrangements for these fast synapses: random or set to specific values (see below). In either case, the fast synapses are not modified as part of the adjustments made to J to get the network to perform a particular task. It is tempting to equate the fast (J_{fast}) and slower (J) synapses in these models to fast AMPA and slower NMDA excitatory synapses or to fast $GABA_A$ and slower $GABA_B$ inhibitory synapses. Although this is correct as far as timescales are concerned, there are issues with this interpretation due to the different ways these two classes of synapses are treated and modified in the models. These remain to be resolved.

The input to the network, $f_{in}(t)$, takes the form of a current injected into each neuron. This current is $f_{in}(t)$ multiplied by a neuron-dependent weight. The vector formed by all N of these weights is denoted by u (**Fig. 1a**; we could also extend these networks to include multiple inputs f_{in} but, for conciseness, we restrict our examples to single-input cases).

The network output is a weighted sum of the normalized synaptic currents generated by all the neurons in the network (**Fig. 2a**, bottom trace; we consider a single output here, but extend this to multiple outputs later). Each network neuron has its own output weight in this sum and, collectively, these weights form an N-component row vector, w (**Fig. 1a**). Output weights are adjusted to minimize the average squared difference between the actual output ws and the desired output f_{out}.

In all of the examples we show, the firing rates of all the network neurons are constrained to realistic values. Another important element in producing realistic-looking spike trains is trial-to-trial variability. Irregular spiking can be generated internally through the random fast synapses we include[36,37] or by injecting a noise current into each network neuron. We do both here.

The spiking networks we discuss come in two varieties that we call rate coding and spike coding. At various points we also discuss what are called rate networks, more abstract models in which network units communicate through continuous variables, not spikes. It is important to keep in mind that the rate-coding case we discuss refers to spiking, not rate, networks.

Driven networks

In any construction project, it is useful to have a working example. As circular as it sounds, one way to construct a network that performs a particular task is by copying another network that does the task. This approach avoids circularity because the example network involves a cheat: it is driven by an input $f_D(t)$ that forces it to produce the desired output (**Fig. 1b**). We call the original network—the one we are constructing (**Fig. 1a**)—the autonomous network (even though it receives the external input f_{in}) and call the example network the driven network (**Fig. 1b**). If there is a single driving input, it is injected into the network neurons through weights described by a vector u_D. Later we will discuss situations in which $P > 1$ driving inputs are used. In this case, u_D is an $N \times P$ matrix. Although the driven network does not receive the original input f_{in} directly, the driving input f_D typically depends on f_{in}, as discussed below. The autonomous and the driven networks contain the same set of fast synapses, but the slower synapses described by the matrix J are absent in the driven network.

The role of the driven network is to provide targets for the autonomous network. In other words, we will construct the autonomous network so that the synaptic inputs to its neurons match those in the driven network. In machine-learning a related scheme is known as target propagation[38,39], and interesting neural models have been built by extracting targets from random networks[40] or from experimental data[41,42].

Obviously, a critical issue here is how to determine the driving input that forces the driven network to perform a task properly. We address this below but, for now, will just assume that we know what the driving input should be. Then, the driven network solves the credit assignment problem for us; we just need to examine what the neurons in the driven network *are* doing to determine what the neurons in the autonomous network *should* do. Even better, the driven network tells us how to accomplish this: we just need to arrange the additional connections described by J so that, along with the term uf_{in}, they produce an input in each neuron of the autonomous network equal to what it receives from the external drive in the driven network. However, there are significant challenges in seeing this program through to completion. (1) We have to figure out what f_D is—in other words, determine how to drive the driven network so that it performs the task. (2) We must assure that this input can be self-generated by the autonomous network with a reasonable degree of accuracy. (3) We must determine the recurrent connection weights J that

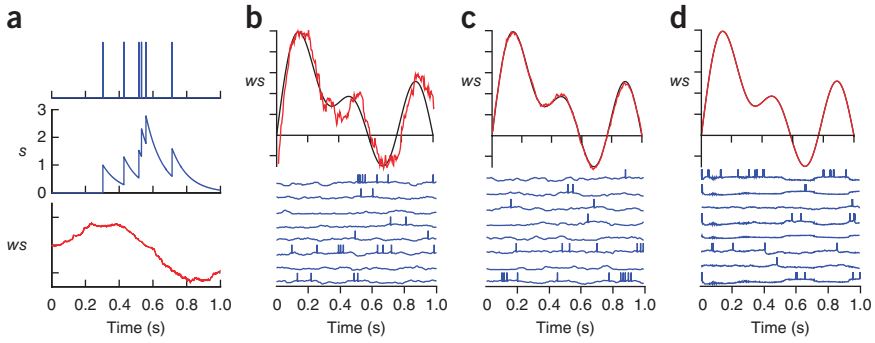

Figure 2 Driven networks approximating a continuous target output. (**a**) Spike train from a single model neuron (top), the normalized synaptic current $s(t)$ that it generates (middle) and the output *ws* computed from a weighted sum of the normalized synaptic currents from this neuron and 99 others (bottom). (**b**–**d**) Results from driven networks with optimally tuned readout weights. In each, the upper plot shows the actual output *ws* in red and the target output f_{out} in black, and the lower plot shows representative membrane potential traces for 8 of the 1,000 integrate-and-fire model neurons in each network. Neurons in the driven network are connected by fast synapses with random weights for **b** and **c** and with weights adjusted according to the spike-coding scheme for **d**. The three panels show the outputs in response to a driving input $f_D = f_{out}$ (**b**), a driving input $f_D = f_{out} + \tau df_{out}/dt$ in a rate-coding network (**c**) and a driving input $f_D = f_{out} + \tau df_{out}/dt$ in a spike-coding network (**d**).

accomplish this task. (4) We must assure that the solution we obtain is stable with respect to the dynamics of the autonomous network. This Perspective covers significant progress that has been made in all four of these areas.

The driven network consists of nonlinear, spiking model neurons connected by either randomly chosen or specifically set (as discussed below) fast synapses (**Fig. 1b**), and the spikes produced by these units are filtered (**Fig. 2a**) and summed (**Fig. 1**) to provide the output. The transformation from the input f_D to the output f_{out} might seem to be quite complex, but it turns out that the effects of nonlinearities and fast network connections can largely be compensated by appropriate choice of the output weights w. In light of this, a first guess for the driving input might be to set $f_D = f_{out}$—that is, to treat the network as if it simply passes a signal from the input to a properly extracted output. This approach can generate good results in rate-based networks[43–47], and it has been tried in spiking networks[7], but in these it only works in limited cases and, in general, poorly (**Fig. 2b**).

A significant advance[6,48] was the realization that the element of the network input-output transformation that cannot be compensated by the choice of output weights is the synaptic filtering at the output, characterized by the time constant τ. Correcting for this synaptic low-pass filtering and its phase delay is easy: we simply define the driving input as a high-pass-filtered, phase-advanced version of the desired output,

$$f_D = f_{out} + \tau \frac{df_{out}}{dt} \quad (1)$$

Using this driving input to produce f_{out} works quite well (**Fig. 2c**). Equation (1), which provides an answer to challenge 1, forms the basis for the work we discuss. Of course, this gives us only a driven version of the network we actually want. In the following sections, we show how to make the transition from the driven network (**Fig. 1b**) to the autonomous network (**Fig. 1a**). Before doing this, however, we introduce an approach that allows the desired output to be produced with greatly enhanced accuracy.

Spike coding to improve accuracy

The network output shown in **Figure 2c** (red line, top panel) matches the target output (f_{out}, black line, top panel) quite well, but deviations can be detected, for example, at the time of the second peak of f_{out}. Some deviations are inevitable because we are trying to reproduce a smooth function with signals $s(t)$ that jump discontinuously every time there is a spike (**Fig. 2a**). In addition, deviations may arise from irregularities in the patterns of spikes produced by the network. The driven network in **Figure 2c** approximates the desired output function because its neurons fire at rates that rise and fall in relation to changes in the function f_{out}. For this reason, we refer to networks of this form as rate coding. Deviations between the actual and desired outputs occur in these networks when a few more or a few fewer spikes are generated than the precise number needed to match the target output. The spike-coding networks that we now introduce[9,10,12] work on the same basic principle of raising and lowering the firing rate, but they avoid generating excessive or insufficient numbers of spikes by including strong fast interactions between neurons. These interactions replace the random fast connections used in the network of **Figure 2b**,**c** with specifically designed and considerably stronger connections (see the Perspective by Denève and Machens[49] in this issue for further discussion of these connections). In general, both excitatory and inhibitory strong fast synapses are required. These synapses cause the neurons to spike in a collectively coherent manner and assure near-optimal performance. For a rate-coding network of N neurons, the deviations between the actual and desired output are of order $1/\sqrt{N}$. In spike-coding networks, these deviations are of order $1/N$, a very significant improvement (as can be seen by comparing the outputs in **Fig. 2c**,**d**). The values of the fast strong connections needed for spike coding were derived as part of a general analysis of how to generate a desired output from a spiking network optimally[9,10]. The use of integrate-and-fire neurons, equation (1) for the optimal input, a determination of the optimal output weights w, and the idea and form of the fast connections are all results of this interesting analysis.

The strength of the fast synapses used in the spike-coding scheme is reflected in the way they scale as a function of the number of synapses that the network neurons receive. Denoting this number by K, one way of assuring a fixed level of input onto a neuron as K increases is to make synaptic strengths proportional to $1/K$. The inability of this scheme to account for neuronal response variability[50] led to the study of networks[36,37] in which the synaptic strengths scale as $1/\sqrt{K}$. Maintaining reasonable firing rates in such networks requires a balance between excitation and inhibition. The fast synapses in spiking-coding networks have strengths that are independent of K, imposing an even tighter spike-by-spike balance between excitation and inhibition to keep firing levels under control.

Spike-coding networks implement the concept of encoding information through precise spiking in a far more interesting way than previous proposals. The spike trains of individual neurons in spike-coding networks can be highly variable (through the injection of noise into the neurons, for example) without destroying the remarkable precision of their collective output. This is because if a spike is missed, or a superfluous one is generated by one neuron, other neurons rapidly adjust their spiking to correct the error.

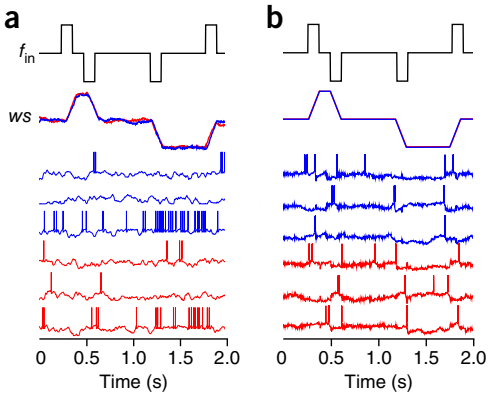

Figure 3 Two autonomous networks of spiking neurons constructed to integrate the input f_{in} (top, black traces). (**a**) A rate-coding network. (**b**) A spike-coding network. For each network, the results from two trials are shown. The upper red and blue traces marked *ws* show the output of the networks on these two trials (they overlap almost perfectly in **b** and are therefore difficult to distinguish), and the bottom blue and red traces show the membrane potentials of three neurons in the networks on the two trials. Note the trial-to-trial variability in the spiking patterns. Each network consists of 1,000 model neurons.

In the following sections, we discuss both spike-coding and rate-coding variants of networks solving various tasks. All of the networks contain fast synapses, but for the rate-coding networks these are random and relatively weak, and their role is to introduce irregular spiking, whereas for the spike-coding networks they take specifically assigned values, are strong and produce precise spiking at the population level. Another important difference is that the elements of the input vector u and recurrent synaptic weights given by J are considerably larger in magnitude for spike-coding than for rate-coding networks.

Autonomous networks

It is now time to build the autonomous network and, to do this, we must face challenges 2–4: how can we arrange the network connections so that the *external* signal f_D that allows the driven network (**Fig. 1b**) to function properly can be produced *internally* and stably by the autonomous network (**Fig. 1a**)? One way to assure that the autonomous network can generate the driving input needed to produce f_{out} is to place restrictions on f_D. Because $f_D = f_{out} + \tau df_{out}/dt$, this also restricts f_{out} and thus limits the complexity of the tasks that the network can perform. We discuss these restrictions and ways to get around them in the following sections.

Because the autonomous network receives the input f_{in} and, if it works properly, produces a good approximation of the desired output f_{out}, one sensible restriction on f_D is to require it to be a linear combination of f_{in} and f_{out}. This imposes the requirement that

$$f_{out} + \tau \frac{df_{out}}{dt} = Bf_{out} + u_R f_{in} \qquad (2)$$

where B and u_R are constants. Because $ws \approx f_{out}$, we can write the current that each neuron in the driven network receives from the driving input, using equation (2), as $u_D f_D \approx u_D Bws + u_D u_R f_{in}$. For the autonomous network to work properly, these currents must be reproduced in the absence of the driving input by the combination of recurrent and input currents $Js + uf_{in}$. Equating the driving and autonomous currents, we see that the autonomous network can be constructed by setting $u = u_D u_R$ and $J = u_D Bw$. This solves challenge 3 (refs. 6,9,10,48).

If $B = 1$, the two terms involving f_{out} in equation (2) cancel, and f_{out} is then proportional to the time integral of f_{in}. The construction we have outlined thus produces, in this case, a spiking network that integrates its input, fairly accurately in the rate-coding case (**Fig. 3a**) and very accurately for the spike-coding version (**Fig. 3b**). Integrating networks had been constructed before the development of the approaches we are presenting[1–5,7,8]; the key advances are that the same methods can be used for more complex tasks and that, in the case of spike coding, accuracy is greatly improved.

For a single function f_{out}, equation (2) can only describe a low-pass filter or an integrator, but a somewhat broader class of functions can be included by extending f_{out} from a single function to a vector of P different functions, while maintaining the restriction that f_D depends linearly on f_{out}. In this extension, B in equation (2) is a $P \times P$ matrix, u_R is a P-component vector, u_D is an $N \times P$ matrix and w is a $P \times N$ matrix. The same approach discussed above, but extended to $P > 1$, allows us to build networks that generate a set of P free-running, damped and/or driven oscillations[6,10,48].

Even with the extension to oscillations, the networks we have discussed thus far are highly limited. This is due to the restriction we placed on f_D by requiring it to be linear in f_{out}. To expand functionality, we must loosen this restriction while continuing to ensure that the autonomous network can generate the signals comprising f_D. Suppose we allow f_D, instead, to be a nonlinear function of f_{out}. In this case, equation (2) is replaced by

$$f_{out} + \tau \frac{df_{out}}{dt} = BH(f_{out}) + u_R f_{in} \qquad (3)$$

where H is a nonlinear function (tanh for example). As in the linear case, we know that $f_{out} \approx ws$, so the driving current into each neuron of the driven network in the nonlinear case is $u_D f_D \approx u_D BH(ws) + u_D u_R f_{in}$. Equating this to the analogous current in the autonomous network, $Js + uf_{in}$, we find that the input weights of the autonomous network are again given by $u = u_D u_R$, but the recurrent circuitry of the network must reproduce the currents given by $u_D BH(ws)$, which, unlike the expression Js, are not linear in s. There are two approaches for dealing with this problem.

The first approach is to modify the spiking neuron model used in the network to include dendritic nonlinearities, meaning that the recurrent input to the neurons of the autonomous network is given by a more complex expression than Js. We implement this by considering the different pieces from which the current $u_D BH(ws)$ is constructed. The term ws can be interpreted as N inputs weighted by the components of w summed on P nonlinear dendritic processes. The function H is then interpreted as a dendritic nonlinearity associated with these processes, and the remaining factor, $u_D B$, describes how the P dendrites are summed to generate the total recurrent synaptic input into the soma of each network neuron. Modifying the neuron model in this way and using a spike-coding scheme, this approach has been developed as a general way to build spiking network models that can be modified easily to perform a wide variety of tasks[22].

The second approach sticks with the original neuron model, uses a rate-coding approach and solves the condition $Js \approx u_D BH(f_{out})$ by a least-squares procedure[2,6,23]. This can work in the nonlinear case because, although the expression Js is linear in s, the normalized synaptic current is generated by a nonlinear spike-generation process. In particular, this process involves a threshold, which supports piecewise approximations to nonlinear functions[2]. To avoid stability problems that may prevent such a solution from producing a properly functioning network, the least-squares procedure used to construct J should be recursive and run while the network is performing the task, using an RLS or FORCE

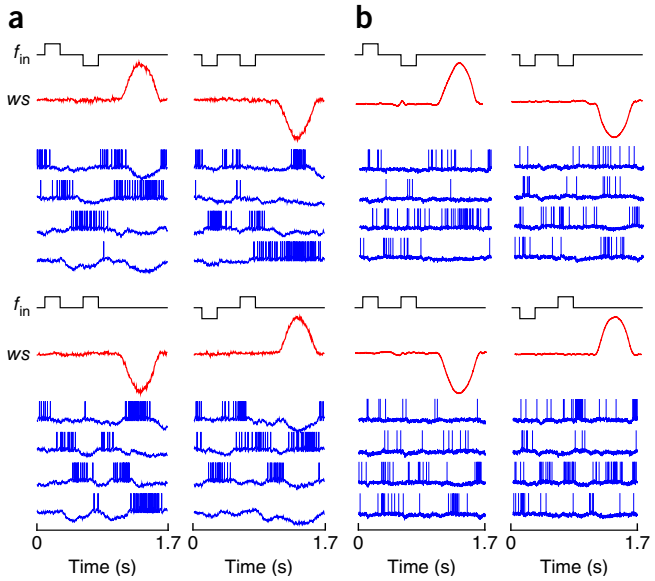

Figure 4 Autonomous networks solving a temporal XOR task. (**a**) A rate-coding network with linear neuronal input integration. (**b**) A spike-coding network with nonlinear neuronal input integration. In both cases, the network output (red traces) is a delayed positive deflection if two successive input pulses have different signs and is a negative deflection if the signs are the same. Blue traces show the membrane potentials of four neurons in the networks.

algorithm[23,44,45]. Although stability is not guaranteed[51], this approach works well in practice, effectively resolving challenge 4.

Figure 4 shows examples of a rate-coding network with linear integrate-and-fire neurons (**Fig. 4a**) and a spike-coding network that includes dendritic nonlinearities (**Fig. 4b**) built according to the procedures discussed in this section and performing a temporal XOR task.

The connection to more general tasks

At this point, the reader may well be wondering what equation (3) has to do with the tasks normally studied in neuroscience experiments. Tasks are typically defined by relationships between inputs and outputs (given this input, produce that output), not by differential equations. How can we use this formalism to construct spiking networks that perform tasks described in a more conventional way? The answer lies in noting that equation (3) defines a *P*-unit rate model, that is, a model in which *P* nonlinear units interact by transmitting continuous signals (not spikes) through a connection matrix *B*. Continuous-variable (rate) networks can perform a variety of tasks defined conventionally in terms of input-output maps if *P* is large enough[43–47]. This observation provides a general method for constructing spiking networks that perform a wide range of tasks of interest to neuroscientists[22,23]. In this construction, the continuous-variable (rate) network plays the role of a translator, translating the conventional description of a task in terms of an input-output map into the differential equation description (equation (3)) needed to construct a spiking network[23]. For the spike-coding network with nonlinear dendrites[22], the continuous variable (rate) model is built into the spiking network, and this allows the network to be quickly and easily readjusted to perform a variety of tasks. The rate-coding networks with linear integrate-and-fire neurons do not require precise dendritic targeting or dendritic nonlinearities, but their recurrent connectivity requires more radical readjustment to allow the networks to perform a new task[23]. In both cases, the power of recurrent continuous variable (rate) networks is used to enhance the functionality of a spiking network.

Discussion

We have reviewed powerful methods for constructing network models of spiking neurons that perform interesting tasks[6,10,22,23]. These models allow us to study how spiking networks operate despite high degrees of spike-train variability and, in conjunction with experimental data, they should help us identify the underlying signals that make networks function.

We have outlined several steps that may be used in the construction of functioning spiking networks, and it is interesting to speculate whether these have analogs in the development of real neural circuits for performing skilled tasks. One step was to express the rules and goals of the task in terms of the dynamics of a set of interacting units described by continuous variables. In other words, the rules of the task are re-expressed in terms of a system of first-order differential equations (equation (3)). It is interesting to ask whether task rules are represented in real neural circuits in the language of dynamics; finding such a representation in experimental data would provide a striking confirmation of the principles of network construction we have discussed. Continuous-variable (rate) networks not only play a key role in the construction of these spiking networks but also describe the fundamental dynamic signals by which the spiking networks operate. This makes them well suited for describing how neural circuits operate, not mechanistically (spiking networks are closer to this) but at a basic functional level.

Our discussion also introduced a driven network that could be used to guide the construction of an autonomous network, and it is interesting to ask whether this step has any biological counterpart. A possible parallel between the driven and autonomous networks we have discussed is the transition from labored and methodical initial performance of a task to automatic and virtually effortless mastery. In the spiking network models, this transformation occurs when an external driving input is reproduced by an internally generated signal. After this transformation takes place, the external signal can be either removed or ignored. Plasticity mechanisms acting within neural circuits may, in general, act to assure that irrelevant signals are ignored and predictable signals are reproduced internally[52–56]. The nature and mode of action of such mechanisms should help us replace the least-squares adjustment of synaptic weights we have discussed with more biophysically realistic forms of plasticity. An alternative might be provided by reward-based learning rules such as reward-modulated synaptic plasticity[57–60].

The spike-coding variants that we have discussed[10,22] are unlikely to operate over a brain-wide scale. Instead, such networks may exist as smaller special-purpose circuits operating with high accuracy. Their predicted experimental signature is strong and dense interconnectivity. The challenge will be to identify the set of neurons that are part of such a circuit. Finally, the nonlinear version of spike-coding networks that we discussed[22] involves both functional clustering of synapses and dendritic nonlinearities. Synaptic clustering has been reported[61–63], but it remains to be seen whether this has the precision needed to support the required dendritic computations. Dendritic nonlinearities of various sorts abound[64,65] and, in this regard, it is important to note that a wide variety of nonlinear functions *H* can support the computations we have discussed.

The ability to construct spiking networks that perform interesting tasks opens up many avenues for further study. These range from developing better methods for analyzing spiking data to studying how large neuronal circuits operate and how different brain regions communicate and cooperate. We hope that future reviewers will be able to cover exciting developments in these areas.

ACKNOWLEDGMENTS

We thank C. Machens, M. Churchland and D. Thalmeier for helpful discussions. Our research in this area was supported by US National Institutes of Health grant MH093338, the Gatsby Charitable Foundation through the Gatsby Initiative in Brain Circuitry at Columbia University, the Simons Foundation, the Swartz Foundation, the Harold and Leila Y. Mathers Foundation, the Kavli Institute for Brain Science at Columbia University, the Max Kade Foundation and the German Federal Ministry of Education and Research BMBF through the Bernstein Network (Bernstein Award 2014).

COMPETING FINANCIAL INTERESTS

The authors declare no competing financial interests.

1. Hansel, D. & Sompolinsky, H. Modeling feature selectivity in local cortical circuits. in *Methods in Neuronal Modeling* 2nd edn. (eds. Koch, C. & Segev, I.) 499–566 (MIT Press, Cambridge, Massachusetts, USA, 1998).
2. Seung, H.S., Lee, D.D., Reis, B.Y. & Tank, D.W. Stability of the memory of eye position in a recurrent network of conductance-based model neurons. *Neuron* **26**, 259–271 (2000).
3. Wang, X.-J. Probabilistic decision making by slow reverberation in cortical circuits. *Neuron* **36**, 955–968 (2002).
4. Renart, A., Song, P. & Wang, X.-J. Robust spatial working memory through homeostatic synaptic scaling in heterogeneous cortical networks. *Neuron* **38**, 473–485 (2003).
5. Song, P. & Wang, X.-J. Angular path integration by moving "hill of activity": a spiking neuron model without recurrent excitation of the head-direction system. *J. Neurosci.* **25**, 1002–1014 (2005).
6. Eliasmith, C. A unified approach to building and controlling spiking attractor networks. *Neural Comput.* **17**, 1276–1314 (2005).
7. Maass, W., Joshi, P. & Sontag, E.D. Computational aspects of feedback in neural circuits. *PLoS Comput. Biol.* **3**, e165 (2007).
8. Burak, Y. & Fiete, I.R. Accurate path integration in continuous attractor network models of grid cells. *PLoS Comput. Biol.* **5**, e1000291 (2009).
9. Boerlin, M. & Denève, S. Spike-based population coding and working memory. *PLoS Comput. Biol.* **7**, e1001080 (2011).
10. Boerlin, M., Machens, C.K. & Denève, S. Predictive coding of dynamical variables in balanced spiking networks. *PLoS Comput. Biol.* **9**, e1003258 (2013).
11. Lim, S. & Goldman, M.S. Balanced cortical microcircuitry for maintaining information in working memory. *Nat. Neurosci.* **16**, 1306–1314 (2013).
12. Schwemmer, M.A., Fairhall, A.L., Denève, S. & Shea-Brown, E.T. Constructing precisely computing networks with biophysical spiking neurons. *J. Neurosci.* **35**, 10112–10134 (2015).
13. Buonomano, D.V. & Merzenich, M.M. Temporal information transformed into a spatial code by a neural network with realistic properties. *Science* **267**, 1028–1030 (1995).
14. Gütig, R. & Sompolinsky, H. The tempotron: a neuron that learns spike timing-based decisions. *Nat. Neurosci.* **9**, 420–428 (2006).
15. Pfister, J.-P., Toyoizumi, T., Barber, D. & Gerstner, W. Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural Comput.* **18**, 1318–1348 (2006).
16. Diesmann, M., Gewaltig, M.-O. & Aertsen, A. Stable propagation of synchronous spiking in cortical neural networks. *Nature* **402**, 529–533 (1999).
17. Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531–2560 (2002).
18. Reutimann, J., Yakovlev, V., Fusi, S. & Senn, W. Climbing neuronal activity as an event-based cortical representation of time. *J. Neurosci.* **24**, 3295–3303 (2004).
19. Vogels, T.P. & Abbott, L.F. Signal propagation and logic gating in networks of integrate-and-fire neurons. *J. Neurosci.* **25**, 10786–10795 (2005).
20. Liu, J.K. & Buonomano, D.V. Embedding multiple trajectories in simulated recurrent neural networks in a self-organizing manner. *J. Neurosci.* **29**, 13172–13181 (2009).
21. Jahnke, S., Timme, M. & Memmesheimer, R.-M. Guiding synchrony through random networks. *Phys. Rev. X* **2**, 041016 (2012).
22. Thalmeier, D., Uhlmann, M., Kappen, H.J. & Memmesheimer, R.-M. Learning universal computations with spikes. Preprint at http://arxiv.org/abs/1505.07866 (2015).
23. DePasquale, B., Churchland, M. & Abbott, L.F. Using firing-rate dynamics to train recurrent networks of spiking model neurons. Preprint at http://arxiv.org/abs/1601.07620 (2016).
24. Memmesheimer, R.-M., Rubin, R., Ölveczky, B.P. & Sompolinsky, H. Learning precisely timed spikes. *Neuron* **82**, 925–938 (2014).
25. Eliasmith, C. *et al.* A large-scale model of the functioning brain. *Science* **338**, 1202–1205 (2012).
26. Hennequin, G., Vogels, T.P. & Gerstner, W. Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron* **82**, 1394–1406 (2014).
27. Ponulak, F. & Kasiński, A. Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. *Neural Comput.* **22**, 467–510 (2010).
28. Florian, R.V. The chronotron: a neuron that learns to fire temporally precise spike patterns. *PLoS One* **7**, e40233 (2012).
29. Brea, J., Senn, W. & Pfister, J.-P. Matching recall and storage in sequence learning with spiking neural networks. *J. Neurosci.* **33**, 9565–9575 (2013).
30. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
31. Mante, V., Sussillo, D., Shenoy, K.V. & Newsome, W.T. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* **503**, 78–84 (2013).
32. Sussillo, D., Churchland, M.M., Kaufman, M.T. & Shenoy, K.V. A neural network that finds a naturalistic solution for the production of muscle activity. *Nat. Neurosci.* **18**, 1025–1033 (2015).
33. Bohte, S.M., Kok, J.N. & Poutré, H.L. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* **48**, 17–37 (2002).
34. Tino, P. & Mills, A.J.S. Learning beyond finite memory in recurrent networks of spiking neurons. *Neural Comput.* **18**, 591–613 (2006).
35. Sporea, I. & Grüning, A. Supervised learning in multilayer spiking neural networks. *Neural Comput.* **25**, 473–509 (2013).
36. van Vreeswijk, C. & Sompolinsky, H. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science* **274**, 1724–1726 (1996).
37. Brunel, N. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *J. Comput. Neurosci.* **8**, 183–208 (2000).
38. LeCun, Y. Learning processes in an asymmetric threshold network. in *Disordered Systems and Biological Organization* (eds. Bienenstock, E., Fogelman, F. & Weisbuch, G.) 233–240 (Springer, Berlin, 1986).
39. Bengio, Y. How auto-encoders could provide credit assignment in deep networks via target propagation. Preprint at http://arxiv.org/abs/1407.7906 (2014).
40. Laje, R. & Buonomano, D.V. Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nat. Neurosci.* **16**, 925–933 (2013).
41. Fisher, D., Olasagasti, I., Tank, D.W., Aksay, E.R.F. & Goldman, M.S. A modeling framework for deriving the structural and functional architecture of a short-term memory microcircuit. *Neuron* **79**, 987–1000 (2013).
42. Rajan, K., Harvey, C. & Tank, D. Recurrent network models of sequence generation and memory. *Neuron* (in the press).
43. Jaeger, H. & Haas, H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science* **304**, 78–80 (2004).
44. Sussillo, D. & Abbott, L.F. Generating coherent patterns of activity from chaotic neural networks. *Neuron* **63**, 544–557 (2009).
45. Sussillo, D. & Abbott, L.F. Transferring learning from external to internal weights in echo-state networks with sparse connectivity. *PLoS One* **7**, e37372 (2012).
46. Lukoševičius, M., Jaeger, H. & Schrauwen, B. Reservoir computing trends. *Künstl. Intell.* **26**, 365–371 (2012).
47. Sussillo, D. Neural circuits as computational dynamical systems. *Curr. Opin. Neurobiol.* **25**, 156–163 (2014).
48. Eliasmith, C. & Anderson, C. *Neural Engineering: Computation, Representation and Dynamics in Neurobiological Systems* (MIT Press, Cambridge, Massachusetts, USA, 2003).
49. Denève, S. & Machens, C. Efficient codes and balanced networks. *Nat. Neurosci.* **19**, 375–382 (2016).
50. Softky, W.R. & Koch, C. The highly irregular firing of cortical cells is inconsistent with temporal integration of random EPSPs. *J. Neurosci.* **13**, 334–350 (1993).
51. Rivkind, A. & Barak, O. Local dynamics in trained recurrent neural networks. Preprint at http://arxiv.org/abs/1511.05222 (2015).
52. Hosoya, T., Baccus, S.A. & Meister, M. Dynamic predictive coding by the retina. *Nature* **436**, 71–77 (2005).
53. Vogels, T.P., Sprekeler, H., Zenke, F., Clopath, C. & Gerstner, W. Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science* **334**, 1569–1573 (2011).
54. Bourdoukan, R., Barrett, D.G.T., Machens, C.K. & Denève, S. Learning optimal spike-based representations. *Adv. Neural Inf. Process. Syst.* **25**, 2294–2302 (2012).
55. Kennedy, A. *et al.* A temporal basis for predicting the sensory consequences of motor commands in an electric fish. *Nat. Neurosci.* **17**, 416–422 (2014).
56. Bourdoukan, R. & Denève, S. Enforcing balance allows local supervised learning in spiking recurrent networks. *Adv. Neural Inf. Process. Syst.* **28**, 982–990 (2015).
57. Potjans, W., Morrison, A. & Diesmann, M. A spiking neural network model of an actor-critic learning agent. *Neural Comput.* **21**, 301–339 (2009).
58. Hoerzer, G.M., Legenstein, R. & Maass, W. Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. *Cereb. Cortex* **24**, 677–690 (2014).
59. Vasilaki, E., Frémaux, N., Urbanczik, R., Senn, W. & Gerstner, W. Spike-based reinforcement learning in continuous state and action space: when policy gradient methods fail. *PLoS Comput. Biol.* **5**, e1000586 (2009).
60. Friedrich, J. & Senn, W. Spike-based decision learning of Nash equilibria in two-player games. *PLoS Comput. Biol.* **8**, e1002691 (2012).
61. Kleindienst, T., Winnubst, J., Roth-Alpermann, C., Bonhoeffer, T. & Lohmann, C. Activity-dependent clustering of functional synaptic inputs on developing hippocampal dendrites. *Neuron* **72**, 1012–1024 (2011).
62. Branco, T. & Häusser, M. Synaptic integration gradients in single cortical pyramidal cell dendrites. *Neuron* **69**, 885–892 (2011).
63. Druckmann, S. *et al.* Structured synaptic connectivity between hippocampal regions. *Neuron* **81**, 629–640 (2014).
64. London, M. & Häusser, M. Dendritic computation. *Annu. Rev. Neurosci.* **28**, 503–532 (2005).
65. Major, G., Larkum, M.E. & Schiller, J. Active properties of neocortical pyramidal neuron dendrites. *Annu. Rev. Neurosci.* **36**, 1–24 (2013).