Bayesian Spiking Neurons II: Learning

Sophie Deneve
sophie.deneve@ens.fr
Group for Neural Theory, Département d'Etudes Cognitives, Ecole Normale Supérieure, College de France 75005 Paris, France

In the companion letter in this issue ("Bayesian Spiking Neurons I: Inference"), we showed that the dynamics of spiking neurons can be interpreted as a form of Bayesian integration, accumulating evidence over time about events in the external world or the body. We proceed to develop a theory of Bayesian learning in spiking neural networks, where the neurons learn to recognize temporal dynamics of their synaptic inputs. Meanwhile, successive layers of neurons learn hierarchical causal models for the sensory input. The corresponding learning rule is local, spike-time dependent, and highly nonlinear. This approach provides a principled description of spiking and plasticity rules maximizing information transfer, while limiting the number of costly spikes, between successive layers of neurons.

1 Introduction

It has long been thought that one goal of learning in neural networks is to find the underlying structure in the sensory world by detecting patterns of correlations (or more generally, dependencies) in the input (Barlow, 2001). Hebbian-style learning rules (Hebb, 1949) find such correlations (Hebb, 1984; Bishop, 1995; Bell & Sejnowski, 1995) and have strong experimental support (Bliss & Collingridge, 1993). However, recent evidence suggests that synaptic long-term plasticity rules depend not only on the average coactivation of the pre- and postsynaptic inputs, but also on the exact temporal structure of their spike trains (Markram & Tsodyks, 1996). This suggests that spikes, and how they are specifically arranged in time, provide the basis for learning interesting structure in the input.

We proposed previously that the basic meaning of a spike is the occurrence of new, unpredictable probabilistic information or, more precisely, an increase in the probability of a particular event. Meanwhile, we proposed that propagation of spikes in cortical networks corresponds to propagation of beliefs in a corresponding Bayesian network (Frey, 1998; Jordan, 1974; Weiss & Freeman, 2001). However, we used a labeled line approach where each neuron was assumed to represent a specific meaning (corresponding to a binary variable). Here, we consider that this selectivity develops, in an unsupervised fashion, from the statistics of the input spike train. Thus, not only the synaptic strength but also the time constant of synaptic integration and the spike generation mechanism are adapted to the spatiotemporal structure of its input. More precisely, Bayesian neurons learn to recognize that a particular subgroup of their synapses tends to switch together from a less active to a more active state and vice versa, with particular probabilities. This is interpreted as caused by an underlying binary variable, switching randomly ON or OFF, and modulating the rates of synaptic input. Consequently, neurons adapt their integrative properties to match how fast the state varies, while the synaptic strengths reflect the coherence of each synapse with this modulation.

The learning rule is directly derived from the Viterbi algorithm (Viterbi, 1967), corresponding to an expectation-maximization algorithm in the context of hidden Markov models (HMMs). Regardless of its biophysical implementation, this learning rule is highly nonlinear and depends on the structure of the input and output spike trains beyond a simple combination of contributions from pairs of input and output spikes. Complex and nonlinear interactions ensure that spikes contribute to learning only when they are highly informative in the context of the entire spike train. This makes a rich set of predictions that can be compared with experimental data.

2 Bayesian Inference in Single Neurons

In this section we briefly summarize the model of Bayesian spiking neuron developed in the companion paper in this issue: "Bayesian Spiking Neurons I: Inference."

2.1 Synaptic Integration.

We considered that individual neurons assume that their synaptic input is generated by a hidden Markov chain (see Figure 1A). Thus, each neuron codes for a particular time-varying binary hidden variable x_t, switching from state 0 to 1 and vice versa with transition probabilities $r_{\text{on}}dt = P(x_{t+dt} = 1|x_t = 0)$ and $r_{\text{off}}dt = P(x_{t+dt} = 0|x_t = 1)$. Second, we considered that the state of the hidden variable causes a collection of N synapses to fire with state-dependent rates. We represented this synaptic input by a vector of binary variable $\mathbf{s}_t = [s_t^i]_{i=1,\ldots,N}$, where $s_t^i = 1$ when the synapse number i fires in the time interval $[t, t+dt)$. Each synapse was activated with a particular probability, $P(s_t^i = 1|x_t = 1) = q_{\text{on}}^i dt$, if the state is 1 and $P(s_t^i = 1|x_t = 0) = q_{\text{off}}^i dt$ if the state is 0.

Inference in this hidden Markov model can be performed by a recurrent process. The log-odds ratio of the hidden state at time t, $L_t = \log\left(\frac{P(x_t=1|\mathbf{s}_{0\to t})}{P(x_t=0|\mathbf{s}_{0\to t})}\right)$, obeys a differential equation in the limit of small dt:

$$\dot{L} = r_{\text{on}}(1 + e^{-L}) - r_{\text{off}}(1 + e^{L}) + \textstyle\sum_i w_i \delta\left(s_t^i - 1\right) - \theta. \tag{2.1}$$

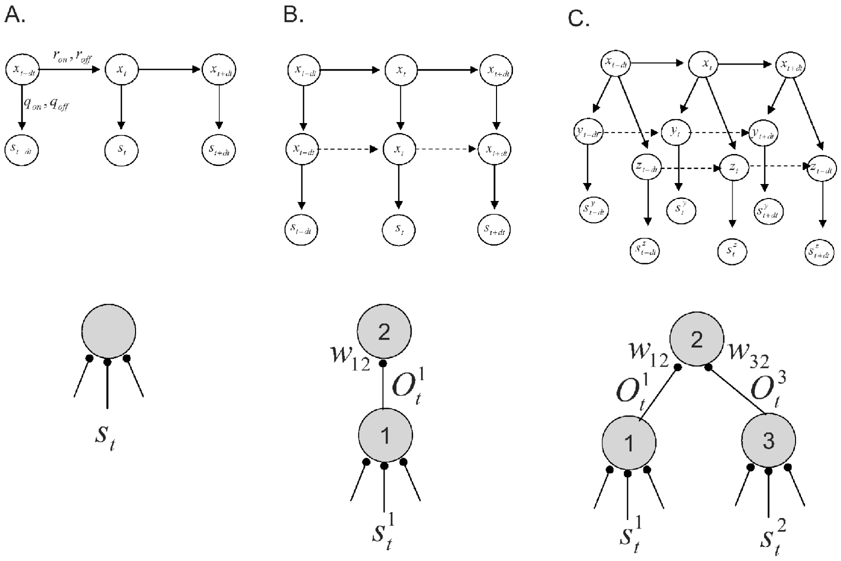

Figure 1: Building and learning hierarchical generative models with neural networks. (A) Bottom: One single neuron learning and inferring about the state of a particular, temporally varying hidden variable. Top: The generative model corresponds to a hidden Markov model (HMM). (B) Two neurons representing two hidden variables, one causing the other (bottom). We suppose that the state of the lowest variable in the hierarchy is independent when one knows its cause, as represented by the dashed arrows in the corresponding generative model (top). (C) Three neurons representing a variable causing two other variables. Only x_t^2 has a temporal dynamics.

w_i, the synaptic weights, describe how informative a synapse i is about the state of the hidden variable, $w_i = \log(\frac{q^i_{\mathrm{on}}}{q^i_{\mathrm{off}}})$. The bias, θ, is determined by how informative it is to receive no spikes, for example, $\theta = \sum_i q^i_{\mathrm{on}} - q^i_{\mathrm{off}}$. The leaky term (the left-hand side of equation 2.1) depends on the transition rates. The synaptic drive, $I_t = \sum_i w_i \delta(s^i_t - 1) - \theta$, is the total contribution of all synaptic events.

2.2 Generation of the Output Spike Train, O_t. We used the same convention for O_t as for s^i_t, for example, $O_t = 1$ when an output spike is fired between time $[t, t + dt)$ and 0 otherwise.

We proposed that a neuron compares online the odds for its hidden variable, L_t, with a prediction G_t computed from its output spike train. A spike is emitted when the odds (a leaky integration of the synaptic input s_t) exceed the prediction (a leaky integration of the output spike train O_t). G_t is indeed what a downstream neuron would obtain as estimates for the log-odds of x_t, $\log\left(\frac{P(x_t=1|O_{0\to t})}{P(x_t=0|O_{0\to t})}\right)$, if it was integrating the output spikes with a synaptic weight of g_o and no bias.

Thus, the dynamical equations relating the input and the output spike trains are:

$$\begin{aligned}\dot{L} &= r_{\text{on}}\left(1+e^{-L}\right) - r_{\text{off}}\left(1+e^{L}\right) + \textstyle\sum_i w_i\delta(s_t^i - 1) - \theta\\ \dot{G} &= r_{\text{on}}\left(1+e^{-G}\right) - r_{\text{off}}\left(1+e^{G}\right) + g_o\delta(O_t - 1)\\ O_t &= 1 \text{ if and only if } L_t > G_t + \frac{g_o}{2}.\end{aligned} \tag{2.2}$$

Here the jump in threshold after a spike, g_o (a positive constant) is the only free parameter, the other parameters being constrained by the statistics of the synaptic input $\mathbf{s}_t$.

3 Learning the Parameters

In this section, we show that the parameters of the generative model, r_{on}, r_{off}, w_i, θ, corresponding respectively to the temporal dynamics, synaptic weights, and bias, can be learned by the neuron from the statistics of its input spike train. This holds for a single neuron receiving Poisson-distributed synaptic inputs, as described in the companion letter in this issue. More important, the same learning rule can be applied to the output spike trains coming from other Bayesian neurons. As a consequence, these neurons can be used as building blocks to represent hierarchies of hidden causes for their input.

The output spike trains of the Bayesian neurons are close to inhomogeneous Poisson processes, whose rates depend on the state of x_t (see section 3 of the companion letter). Thus, we propose to apply a learning rule designed for HMMs with Poisson input to estimate the statistics of the hidden variable x_t from the output spike train O_t. This corresponds to approximating the output spikes as temporally independent observations of the hidden state. This will necessarily be suboptimal, since there are some temporal dependencies, albeit small, between the output spikes. However, we show through simulations that the information loss resulting from this approximation is minimal and depends critically on the postspike jump g_o.

We first describe the "exact" learning rule from a model consisting of a single neuron with Poisson-distributed input, as described in the previous section (see Figure 1A). We then proceed to show that model neurons can also learn the proper parameters when their inputs come from other Bayesian neurons rather than purely Poisson-distributed synaptic inputs.

3.1 Learning in Single Neurons with Poisson Input. The parameters of a hidden Markov chain (see Figure 1A) can be learned by an expectation-maximization (EM) algorithm, a variant of the Viterbi algorithm (Viterbi, 1967). This algorithm finds the parameters maximizing the probability of observing the stream of noisy data given the model. This is done by alternating an expectation step, where the expected value of x_t is computed given the observations and the current set of parameters, and a maximization step, which updates the parameters so as to maximize the likelihood of the observations given the expected value of x_t.

Here we propose a version of the EM algorithm that can be implemented by a single neuron: at each time, the neuron updates the sufficient statistics of its hidden variable according to its new synaptic input (expectation step) and updates the synaptic weights and transition rates accordingly (maximization step). Thus, expectation and maximization steps are not temporally separated, and the parameters are updated online without requiring a memory for long sequences of input data.

3.1.1 Expectation Step. For the expectation step, several sufficient statistics are computed online from the observed synaptic input, given the current set of weights and transition rates (see the letter appendix). Values of interest are the expected state, or time spent in the ON state, $\tau_{\text{on}}(t) = \int_{t-\tau}^{t} P(x_u = 1|\mathbf{s}_{0\to t})du$; the expected number of ON→OFF transitions $N_{\text{on}\to\text{off}}(t) = \int_{t-\tau}^{t} P(x_u = 1, x_{u+du} = 0|\mathbf{s}_{0\to t})du$; one spike-triggered expectation per synapse (the expected number of input spikes while in the ON state) $N^i_{\text{on}}(t) = \int_{t-\tau}^{t} \delta(s^i_u - 1)P(x_u = 1|\mathbf{s}_{0\to t})du$; and the mean number of input spikes from each synapse $N^i = \int_{t-\tau}^{t} \delta(s^i_u - 1)du$. τ represents a finite sliding temporal window on which these expected values are computed (see the appendix for a more rigorous description). We propose that for the sake of learning the statistics of its input, each neuron estimates two global averages and two local spike-triggered averages per synapse. $\tau_{\text{on}}(t)$, $N_{\text{on}\to\text{off}}(t)$, and $N^i_{\text{on}}(t)$ can be derived recurrently from the synaptic input in the same way as L_t and G_t.

3.1.2 Maximization Step. In standard (batch) implementations of EM, parameters are updated once the expectation step has been completed on a suitably long temporal window. In neurons, we propose that expectation and maximization steps occur simultaneously and online, the weights and transition rates being updated according to the online expected values. One can show that this algorithm converges toward the estimate for batch EM (see the appendix).

Thus, the sufficient statistics for x_t, computed online from the synaptic input, are used to update the parameters as follow:

$$w_i(t) = \log\left(\frac{N^i_{\text{on}}(t)}{N^i(t) - N^i_{\text{on}}(t)}\right) - \log\left(\frac{\tau_{\text{on}}(t)}{1 - \tau_{\text{on}}(t)}\right)$$

$$\theta(t) = \sum_i \frac{N^i(t) - N^i_{\text{on}}(t)}{1 - \tau_{\text{on}}(t)} - \frac{N^i_{\text{on}}(t)}{\tau_{\text{on}}(t)}$$

$$r_{\mathrm{on}}(t) = \frac{N_{\mathrm{on}\to\mathrm{off}}(t)}{1 - \tau_{\mathrm{on}}(t)}$$
$$r_{\mathrm{off}}(t) = \frac{N_{\mathrm{on}\to\mathrm{off}}(t)}{\tau_{\mathrm{on}}(t)} \tag{3.1}$$

More intuitively, learning a weight corresponds to estimating how much more (or less) probable than average was x_t when a spike was received from that synapse. Thus, the weights are positively or negatively incremented depending on whether the probability of x_t tends to be larger or smaller than its running average at the moment of the synaptic input. Similarly, learning the bias corresponds to estimating how much more (or less) probable than average was x_t when no spike is received. Both of these rules are Hebbian since learning occurs when presynaptic spikes correlate with postsynaptic activity. Learning the transition rates corresponds to estimating how often x_t switches state. In effect, this amounts to computing temporal derivatives of the expected state.

3.2 Learning in Networks. So far, we have considered learning in a Bayesian neuron faced with the problem of estimating the statistics of an underlying hidden variable from its Poisson distributed synaptic inputs. If we want to be able to use this neuron as a building block to represent another level of hierarchy, we need to show that the same learning rules can be applied with success to learn the statistics of hidden variables from the Poisson-like output spike trains of other Bayesian neurons. The problem here is that even if the spiking dynamics ensures that the output spike train is as close as possible from a Poisson process, it is not usually perfectly Poisson (see the companion letter). Small deviations from perfect independence between interspike intervals might be sufficient to induce strong deviations between the true and the learned parameters, and thus hinder the transmission of information between layers. In this section, we show through simulations that this does not appear to be the case. Thus, the output spike trains of Bayesian neurons can indeed be treated as Poisson for all practical purposes.

3.2.1 Learning a One-to-One Connection Between Two Bayesian Neurons. As a first step, we consider the case when a Bayesian neuron (neuron 2) receives input from another Bayesian neuron, (neuron 1) (see Figure 1B). The corresponding generative model consists of two coupled hidden Markov chains, where the top variable x^2 (the cause) ignores the temporal dynamics of the bottom variable (x^1). This is represented by the fact that temporal prediction arrows from x^1_t to x^1_{t+dt} are dashed in Figure 1B. This specifically means that the state of x^1 at time t does not depend on the state at time $t - dt$ if one knows the state of x^2_t (x^1_t is conditionally independent of the past). This corresponds, for example, to the case when an object causes the presence of a particular feature: the visual feature at time t depends on the presence of the object at time t, but not on whether this feature was present at $t - dt$, since the feature does not exist by itself. (For a detailed discussion, see the companion letter.)

Moreover, we suppose that in this very simplified situation, x^2 has no consequence other than x^1 (i.e., neuron 2 receives no input other than neuron 1), and x^1 has no other cause than x^2 (i.e., neuron 1 sends a connection to neuron 2 only). As a consequence, the two neurons will learn to represent the same hidden variable $x_1 = x_2 = x$ (see Figure 1B).

Neuron 1 learns the parameters describing the statistics of x_t from its Poisson input $\mathbf{s}_t$ using the EM algorithm. After learning, its log-odds ratio L_t^1 corresponds to the probability of x_t given the past synaptic inputs $\mathbf{s}_{0\to t}$. Neuron 2, applying the same learning rule on the output spike train O_t^1, learns a weight w_{12}, a bias θ_2, and transition rates r_{on}^2 and r_{off}^2, and estimates the log-odds ratio of x_t by the following differential equation:

$$\dot{L}_t^2 = r_{\text{on}}^2(1 + e^{-L_t^2}) - r_{\text{off}}^2(1 + e^{L_t^2}) + w_{12} O_t^1 - \theta_2. \tag{3.2}$$

Applying the EM algorithm on the output spike train and using this equation to infer the probability of x_t given past inputs corresponds to considering that the spike train O_t^1 provides temporally independent evidence for x_t, that is, O_t^1 depends on x_t but not on O_{t-dt}^1 (i.e., O_t^1 is an inhomogeneous Poisson process whose rate depends on the state). In order to show that this approximation is correct, we looked at the discrepancy between the true and the learned probabilities. Indeed, L_t^2 can be considered an estimate of the true probability L_t^1 or, alternatively, as a decoding of the output spike train O_t^1. A good match between L_t^1 and L_t^2 would provide evidence that the learning algorithm (designed for Poisson inputs) is efficient even when applied on the output spike trains of Bayesian neurons, and thus, that approximating O_t^1 as a Poisson process is not detrimental. This is also a measure of how efficient the encoding of the log-odds L_t^1 is by the output spike train O_t^1.

Examples of L_t^1 and L_t^2 temporal profiles are plotted in Figure 2. The solid line corresponds to the true log-odds L_t^1, the thick dotted line to the estimated log-odds L_t^2, and the thin dotted line (visible only on Figure 2A) to the true log-odds, L_t, computed directly from the parameters of the generative model. In order to learn w_{12} (corresponding to the jumps of thick dotted line after a spike), the bias θ_2, and the transition rates r_{on}^2 and r_{off}^2, we sampled x_t and the synaptic input $\mathbf{s}_t$ according to the statistics of the HMM, and applied the neural dynamical equation (see equation 2.2). The output spike train O_t^1 was then used as a training sequence for the learning algorithm (see equation 3.1). After convergence of the parameters (i.e., when they fluctuate by less than 1% of their value during more than 10τ), we applied equation 3.2 to get L_t^2. In this illustrative example, we used

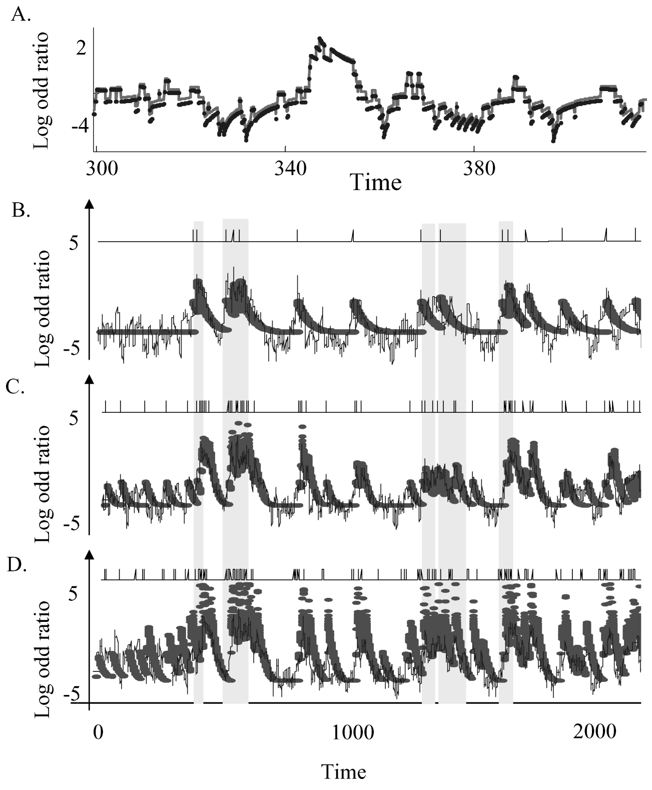

Figure 2: Learning and decoding performance as a function of the jump in threshold, g_0. (A) Log-odds L_t^1 (solid line) computed from the synaptic input s_t with the parameters learned by neuron 1. The log-odds after learning closely matched the true log-odds L_t(dotted line) computed with the true parameters of the HMM. (B) A neuron (neuron 2) can learn to decode the log-odds ratio of another neuron (neuron 1), L_t^1 (blue solid line). To do so, it needs to learn w_{12} and θ_2 from neuron 1's output firing rate O_t^1 (horizontal lines). The resulting estimate, L_t^2 (thick grey dots), closely matches the original log probability ratio. In this example, $g_0 = 4$ for neuron 1. (C) Same as B for $g_0 = 1.5$. (D) Same as B for $g_0 = 0.5$. For larger g_0, the output spike train become sparser (i.e., with fewer spikes) at the cost of neglecting small fluctuations in probability. The shaded area represent time when $x_t = 1$.

arbitrarily chosen parameters: $r_{\text{on}}^1 = 0.001, r_{\text{off}}^1 = 0.01$, 50 synapses to neuron 1 with $q_{\text{on}}^i = 0.03$(corresponding to 30 spikes per seconds), $q_{\text{off}}^i = 0.02$, $dt =$ 0.1 ms, 30 synapses to neuron 1 with $q_{\text{on}}^i = 0.02$, $q_{\text{off}}^i = 0.03$, $\tau = 100{,}000dt$ (10 seconds).

By repeating the procedure 100 times with different initial random settings of the parameters, we found that the parameters learned by neuron

1 closely matched the true parameters of the HMM: $\hat{q}^1_{\text{on}} = 0.03 \pm 0.005$, $\hat{q}^1_{\text{off}} = 0.02 \pm 0.004$, $\hat{r}^1_{\text{on}} = 0.001 \pm 0.0006$, $\hat{r}^1_{\text{off}} = 0.01 \pm 0.003$. As a consequence, the log-odds L^1_t (computed by neuron 1 using the parameters learned by the online EM algorithm) closely matched the true log-odds L_t (computed using the true parameters of generative model). This is illustrated in Figure 2A.

Figure 2B illustrates the match between L^1_t and L^2_t when $g_o = 4$ for neuron 1. Note that on average, neuron 1 receives about 1000 spikes in a second from its synaptic input $\mathbf{s}_t$ but fires, on average, seven spikes per second. Despite this high level of compression, the log-odds computed by neuron 2 (L^2_t) closely matches the log-odds computed by neuron 1 (L^1_t).

3.2.2 g_o and the Trade-Off Between the Quality and Cost of the Code. We used this simple model to investigate in more detail the role of the postspike jump in threshold g_o. g_o represents the increase in the probability for $x_t = 1$ that is required for the neuron to decide to fire an output spike. As we have seen in the companion letter, Bayesian neurons are analogous to leaky integrate-and-fire neurons, where g_o corresponds to the reset in membrane potential after a spike. g_o is the only free parameter that is not constrained by the statistics of x_t, and it directly sets the input-output gain of the neuron, since the output rate is approximately $\bar{O} = \frac{1}{g_o}[\bar{I}]^+$ (see the companion letter).

We varied g_o in neuron 1 from 4 (see Figure 2B) to 1.5 (see Figure 2C) and 0.5 (see Figure 2D). For better comparison, we used the same synaptic input $\mathbf{s}^1_t$ to neuron 1 for producing the three plots. Thus, the three panels correspond to the neural responses, after training, to the same exact input, and only g_o varies. The log probability ratio of neuron 1, L^1_t, is thus identical in the three panels. What varies is the output spike train fired by neuron 1, O^1_t (thick vertical lines), and the log-odds L^2_t estimated from this output spike train by neuron 2, as well as the weight and bias learned by neuron 2, w_{12} and θ_2. We found in our simulation $w_{12} = 2.8$ and $\theta_2 = -0.06$ for $g_o = 0.5$, $w_{12} = 2.5$ and $\theta_2 = -0.02$ for $g_o = 1.5$, and $w_{12} = 2.13$ and $\theta_2 = -0.005$ for $g_o = 4$.

As g_o increases, the number of spikes fired by neuron 1 decreases. The output representation in Figure 2B corresponding to $g_o = 4$ (on average, 7 spikes per second, or approximately 1 output spike for 150 input spikes) is much sparser than in Figure 2D corresponding to $g_o = 0.5$ (on average, 50 spikes per second—i.e., 1 output spike for 25 input spikes). This is because the output firing rate is approximately proportional to the inverse of g_o.

The cost of saving spikes is to lose precision in the representation of L^1_t. Thus, small fluctuations in probability are neglected. The match between real and decoded log-odds degrades as g_o gets larger. However, the match is still remarkably good even for very large g_o.

We can conclude from this example, confirmed by numerous simulations performed with different sets of parameters, that the model is remarkably efficient in recovering a good estimate of the log-odds from an extremely sparse output spike train. In fact, the neural dynamics and spike generation rules ensure that output spikes are fired at the right time in order to represent, as sparsely as possible, the log-odds ratio. g_o, the postspike jump, controls the cost-accuracy trade-off—the cost being the number of spikes and the accuracy the representation of small fluctuations in probability of the hidden state.

3.2.3 Multiple Layers. The ultimate goal of our approach is to be able to use the model neuron described in the companion letter as a building block for constructing progressively more complex Bayesian networks or progressively more elaborate hierarchical causal models for the sensory input, motor output and behavioral tasks. A complete proof of this claim goes beyond the scope of this letter; however, we can already verify that the probabilistic evidence can be correctly transmitted in a restricted family of Bayesian networks.

This restricted family is described in the companion letter. It corresponds to hierarchical trees of binary variables, where only the variables at the top of the hierarchy have a temporal dynamics of their own (see Figure 1C). Inference in this restricted family corresponds to propagating beliefs from the consequences (e.g., the visual features, such as stripes) to the causes (e.g., the objects causing these features, such as tigers). Here we consider only this form of processing (feedforward). Feedback processing, corresponding to propagating beliefs from the causes to the consequences, will be considered in future work.

Inference in these models is implemented by a neural network with a treelike structure that matches the structure of the underlying generative model (see Figures 1C, 3A, and 3C). Here, different layers of neurons correspond to different layers of causality. We use the online EM algorithm described previously to learn the weights of the connections between two layers and the temporal dynamics (transition rates) of each neuron. Each neuron acts as an information filter, learning to transmit synaptic events caused by a slowly varying hidden state (i.e., correlated) and to discard other (uncorrelated) synaptic events. This ensures that the neuron transmits most of the useful information to the next layer. However, due to the overly convergent nature of these networks, this information will be concentrated in much fewer spikes on the top layers, and fluctuations in synaptic inputs lacking interesting correlations will be lost.

In Figures 3 and 4, we describe the learning performance in a particular example. In this very simple "world," 4-pixel-long horizontal bars can appear and disappear at random nonoverlapping locations on a 3×12 retina ($r_{\text{on}}^{\text{bar}} = 0.05$, $r_{\text{off}}^{\text{bar}} = 0.1$ at each of the nine possible locations). The presence of a bar modulates the firing rates of lateral geniculate nucleus (LGN) units whose receptive fields overlap with these bars (light shaded unit, $q_{\text{on}}^{i} = 0.08$, $q_{\text{off}}^{i} = 0.02$; dark shaded units, $q_{\text{on}}^{i} = 0.04$, $q_{\text{off}}^{i} = 0.075$). Moreover, horizontal bars can also be caused by the presence of a 12-pixel-long horizontal

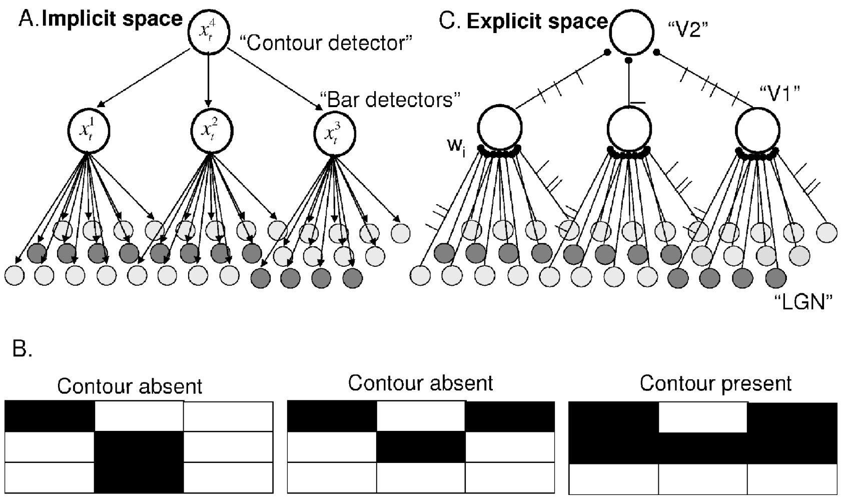

Figure 3: The generative model and the corresponding neural network for a toy world composed of bars and contours. (A) Generative model used to generate the spiking input s_t (from the "retina" to the "LGN"). On a 3×12 retina, a contour consists of three nonoverlapping horizontal bars, each 4 pixels long, aligned on the center. The presence of this contour ($x_t^4 = 1$) results in the presence of horizontal bars at the corresponding locations. The contour appears and disappears randomly. To make the task more difficult, this contour is embedded in "noise," that is, bars can also appear independently and randomly at any of the nine possible nonoverlapping locations. In turn, these horizontal bars result in a particular probability of emitting a spike for LGN units (shaded units). For clarity, only three of the nine possible horizontal bars (V1 units) are represented. (B) Three possible images generated by the generative model. As contours and bars can appear and disappear over time, these three panels can be thought of as frames of a movie. (C) Corresponding spiking neural network. LGN cells, detecting light or dark pixels, connect to V1 cells detecting horizontal bars, which themselves connect to V2 contour detectors.

contour. The contour is composed of three bars appearing at collinear locations in the center of the retina. These contours appear and disappear randomly ($r_{\text{on}}^{\text{contour}} = 0.005$, $r_{\text{off}}^{\text{contour}} = 0.01$). The interaction between contours and spontaneously generated bars corresponds to a nonexclusive "or" relationship.

We used the corresponding generative model (see Figures 3A and 3B) to generate lateral geniculate nucleus (LGN) spike trains. These LGN spike trains were then used as a training set for a three-layer neural network (see Figure 3C).

For all neurons, the initial weights w_i from the LGN to the V1 layer and from the V1 layer to the V2 neuron were sampled from a normal distribution, the biases were set at $\theta_i = 0$, and the initial transition rates

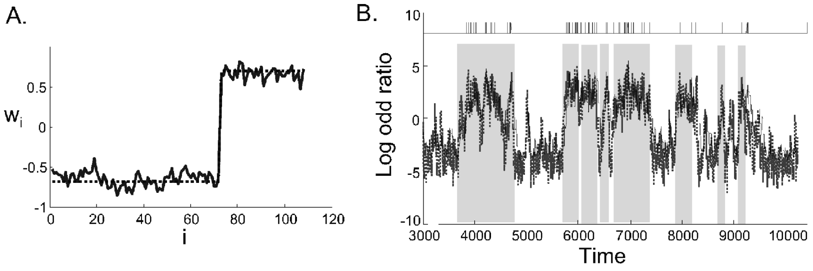

Figure 4: Learning performance on a multilayer network of Bayesian neurons. (A) Weights from the LGN layer to V1 bar detectors, learned by the approximate online EM algorithm (solid line). These weights are close to the real weights predicted by the generative model (dotted line). (B) Log probability ratio of the V2 contour detector neuron on one particular trial (dark solid line) together with its output spike train (thick vertical lines). Light dotted line: Performance on an ideal observer (see the text). Shaded areas indicate periods when the contour was present.

were set at $r_{\text{on}} = r_{\text{off}} = 0.1$ for both bar neurons and the contour neuron. The connection strengths, biases, and transition rates for the V1 and V2 neurons were then learned using the online EM algorithm (see equation 3.1), with $\tau = 100{,}000dt$ and $dt = 0.1$ ms, using solely the LGN spike train generated previously. Learning was stopped when the parameters fluctuated by less than 5% of their value for 10τ. This learning process was repeated 150 times with different initial settings of the parameters to verify the convergence and stability of the learning algorithm.

Approximately half of the bar neurons became "off neurons" in the sense that they represented the absence, rather than the presence, of a bar. Depending on the initial random settings of the parameters, the top V2 neuron learned to detect either the presence or the absence of the contour ($x_t^4 = 1$ in Figure 3A could correspond to "contour present" or "contour absent"). This does not affect the performance of the network, since $p(x_t^4 = 1) = 1 - p(x_t^4 = 0)$.

We found a good match between the "real" synaptic weights (defined by the generative model) and the learned synaptic weights (see Figure 4A). The estimated transition rates were noisier, since they are second-order statistics, which are harder to estimate from streams of noisy data. However, on multiple repetition of the learning procedure with different initial parameters, the estimated transition rates were unbiased ($\hat{r}_{\text{on}}^{\text{contour}} = 0.005 \pm 0.0005$, $\hat{r}_{\text{off}}^{\text{contour}} = 0.01 \pm 0.004$, $\hat{r}_{\text{on}}^{\text{bar}} = 0.05 \pm 0.01$, and $\hat{r}_{\text{off}}^{\text{bar}} = 0.1 \pm 0.04$ for bars not belonging to the contour).

We compared the performance of the network with the performance of an ideal observer for detecting the contour (see Figure 4B). To obtain the ideal observer performance, we performed exact online inference directly on the LGN input, taking the LGN inputs as a noisy observation of a contour, thus bypassing the bar layer. As illustrated in Figure 4B, the ideal observer outperforms the network only marginally.

Note that we used a generative model for the sensory data (i.e., the LGN spikes) with several coexisting temporal dynamics: both the bars and the contour can appear and disappear on their own. Thus, taken as a whole, this generative model violates the restriction that we put at the beginning: that only the object at the top of a hierarchy should have a temporal dynamic. However, neurons in the corresponding neural network receive no feedback connections, and thus they ignore the existence of variables that are higher in the hierarchy. From the point of view of each neuron, its hidden variable x_t has a temporal dynamic and is the highest in the hierarchy, while its log-odds L_t corresponds to the posterior probability of x_t when considering only a subpart of the generative model corresponding to x_t and all its children.

One might wonder why we had to go through the V1 local bar detector at all in order to compute the probability of a contour. This probability could be computed directly from the LGN input. However, the number of synaptic contacts from the LGN to the V2 cell needed for such a computation is very large. One of the merits of the hierarchical model is to recode relevant probabilistic information into a sparse format, so that many weakly informative LGN spikes can be recoded in a few much more informative V1 spikes. Thus, the number of synapses can be kept to a reasonable level.

Even more important, more elaborate generative models (that is, more complicated worlds) will not allow such short-cuts. For example, local bars could be caused by many different events, such as contours, textures, or shadows. These different potential causes would need to compete in order to select the real underlying causes of the sensory input. Also, contours can increase the probability of the presence of a local edge when the available local LGN information would be insufficient for detecting this edge. Feedback from V2 to V1 or lateral connections within V1 could be used to clean up the noise and solve ambiguities by sharing probabilistic information between collinear edges arranged along a likely contour. Lateral and feedback interactions have a crucial role for perceptual tasks that cannot be represented by a treelike generative model as in Figure 3. Image segmentation is an example of sensory processing that cannot be purely bottom up (Lee & Mumford, 2003). Our preliminary work shows that these computations can be successfully implemented and learned with recurrently connected Bayesian spiking neurons, with feedforward, feedback, and lateral connections, allowing the implementation of hierarchical causal model of the sensory inputs or the tasks. However, these go beyond the scope of this letter, focusing on single neurons as the building blocks of these more complex networks. The network computations will be extended in future work.

4 Prediction for Spike-Time-Dependent Plasticity

The learning rules governing plasticity depend on the temporal structure of the input spike trains received at all synapses. Testing directly the prediction of the model for synaptic plasticity is difficult, since all the synaptic input received by a cortical neuron will never be directly available experimentally. Typically one can record a single input spike train s_t^i and the output spike train O_t.

Fortunately, the output spike train of the model neuron, O_t, directly reflects the global belief state of the neuron. We have already seen that a good estimate of $p(x_t|\mathbf{s}_{0\to t})$ can be recovered from O_t (see Figure 2). Thus, one can predict the changes induced in synaptic strength just by looking at the input spike train at a single synapse and the output spike train. Similarly, one can predict the change in transition rates r_{on} and r_{off} just by looking at the output spike trains. Note that we do not claim here that learning is based on the output spike train (the online EM algorithm presented here is applied strictly on the input spikes).

For these reasons, we chose to express our predictions for synaptic plasticity in terms of input and output spike-times, since output spikes are a direct expression of the log-odds of x_t, are easily measurable, and describe the true output of the neuron.

The next section describes how the temporal profile of the odds for x_t is reconstructed from the output spike train in a temporal window of length T, $O_{0\to T}$ (see Figure 5A).

4.1 Predicting the Odds of x_t from the Output Spike Train. If we consider that the output spikes are temporally independent observations for the state x_t, then the probability of the hidden state, $P(t) = P(x_t|O_{0\to T})$, is a product of the contribution from past and future output spikes, $P(t) = \frac{1}{Z}P_b(t)P_f(t)$, where $P_f(t) = P(x_t|O_{0\to t})$ and $P_b(t) = P(O_{t+dt\to T}|x_t)$, and Z is a normalization constant. To keep up with our previous notation, the log-odds ratio of x_t, L_t^{tot}, is given by $L_t^{\mathrm{tot}} = L_t^f + L_t^b$, the sum of the forward log-odds, $L_t^f = \log(\frac{P^f(t)}{1-P^f(t)})$, and backward log-odds $L_t^b = \log(\frac{P^b(t)}{1-P^b(t)})$.

The forward log-odds are equivalent to the log-odds ratio L_t, considered previously, but computed from the output spike train:

$$\dot{L}^f = r_{\mathrm{on}}(1+e^{-L^f}) - r_{\mathrm{off}}(1+e^{L^f}) + w_o O_t - \theta_o. \tag{4.1}$$

This is equivalent to another neuron's estimate of L_t based on the output firing rate (see the grey dots in Figure 2). w_o is the output weight, and θ_o the output bias.

The backward log-odds L_t^b represent what is known about the initial state after observing a stream of subsequent output spikes. It is computed by integrating the inputs, but backward rather than forward in time. Taking

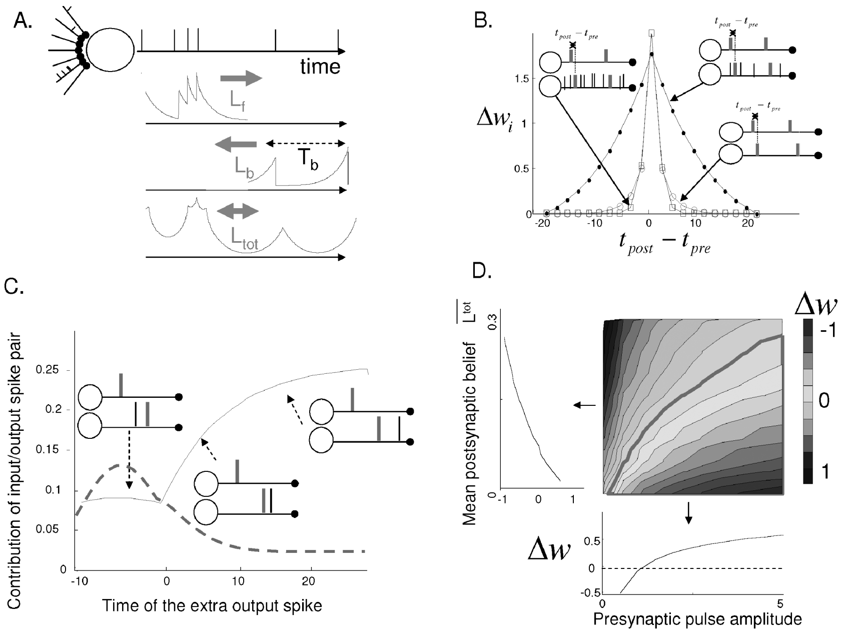

Figure 5: Spike-time-dependent learning implementing expectation-maximization. (A) The forward and backward expectations for the hidden state can be estimated from the output spike train by propagating evidence forward and backward in time (see the text). (B) The resulting spike-time-dependent plasticity rule is nonlinear and depends on the structure of the output spike train. Here we plot the predicted synaptic changes in response to repeated coupling of input and output spikes as a function of the delay between them. The synaptic weight is enhanced when the input and output spikes are coincident within a particular temporal window. This temporal window, however, depends on the detailed structure of the output spike train. The different curves correspond to different firing rates of the postsynaptic neuron (not counting the training pairs). Circles: zero firing rate. Dots: firing rate equal to $-\theta_o$. Squares: firing rate equal to $-\theta_o + 0.01$. (C) Contribution of an output (postsynaptic) spike (spike 1) occurring immediately after an input (presynaptic) spike as a function of the timing of another output spike (spike 2). The contribution is defined as the difference between the weight when spike 1 is present and when spike 1 is absent, for identical timing of the presynaptic spike and spike 2. Solid line: output firing rate at $-\theta_o$. Dashed line: output firing rate at 0. (D) Changes in weight in response to paired input-output pulses of activity (see the text). The change in synaptic weight is plotted as a function of the mean belief for $x_t = 1, \bar{L}_{tot}$, and the amplitude of the input pulses.

the limit for small dt (see appendix A in the companion letter), the function $H_T(t) = L^b_{T-t}$ obeys the following differential equation:

$$\dot{H}_T = r_{\text{on}}(1 - e^{H_T}) - r_{\text{off}}(1 - e^{-H_T}) + w_o O_t - \theta_o. \tag{4.2}$$

The output weight, w_o, is a positive constant, corresponding to a transient increase in the probability of $x_t = 1$ reflected by the output spike. The bias θ_o is a (small) positive constant, corresponding to a decrease in the probability of $x_t = 1$ reflected by the absence of an output spike. These parameters can be learned, together with the transition rates r_{on} and r_{off}, by online expectation maximization applied to O_t (it is equivalent to learning r^2_{on}, r^2_{off}, w_{12}, and θ_{12} in section 3.2). The log-odds computed in this way are a good match to the true log-odds computed from all input spike trains (see section 3.2, Figure 2). For all practical purposes, we chose $r_{\text{on}} = 0.001$, $r_{\text{off}} = 0.01$, $w_o = 1$, and $\theta_o = -0.01$. The exact values of these parameters do not affect the qualitative results reported here.

Once the probability of x_t has been computed from the output spike trains, we can use it to estimate the new weights and the new transition rates, using the standard Viterbi algorithm. In particular, the new weights w^i_{new} are given by

$$w^i_{\text{new}} = \log\left(\frac{\bar{P}^i}{\bar{s}^i - \bar{P}^i}\right) - \log\left(\frac{\bar{P}}{1 - \bar{P}}\right), \tag{4.3}$$

where

$$\bar{P} = \int_0^T P(t)dt \tag{4.4}$$

$$\bar{P}^i = \int_0^T \delta(s^i_t - 1)P(t)dt \tag{4.5}$$

$$\bar{s}^i = \int_0^T \delta(s^i_t - 1)dt. \tag{4.6}$$

In the following simulations, we chose an initial weight of $w_i = 0.5$.

4.2 Learning as a Function of Input-Output Spike Synchrony. We have seen that synaptic weights are potentiated or depressed depending on whether the probability of $x_t = 1$ conditioned on an input spike is larger or smaller than its expected value. Output spikes express an increase in probability for $x_t = 1$. Thus, the weights will be increased or decreased as a function of the synchrony between input and output spikes. Figure 5B describes the predicted change in synaptic weight (from an initial input weight $w_i = 0.5$) in response to 100 repeated input-output spike pairs, spaced by 100 ms, as a function of the delay between input and output spikes. We estimated the log-odds L^{tot}_t from the output spike train using the procedure described in the previous section (see Figure 5A). More exactly we applied equations 4.1 and 4.2 to the output spike train O_t.

4.3 Dependence on the Output Firing Rate. The synaptic weight is increased when input and output spikes are coincident within a particular temporal window. However, the size of this temporal window greatly depends on the level of activity of the postsynaptic neuron. Figure 5D illustrates the change in weight following 100 repeated pairings of input and output spikes as a function of the delay between them. Successive pairs are spaced by 200 ms. Moreover, we suppose that the postsynaptic neuron also fires additional spikes at random times (according to a Poisson process) in addition to the spikes belonging to the training pairs. The learning window is the longest (i.e., learning is the most efficient) when the postsynaptic firing rate is around $-\frac{\theta_o}{w_o} = -\theta_o$. In this case, the postsynaptic neuron's average log-odds is kept at the level of the prior $L_o = \log(\frac{r_{\text{on}}}{r_{\text{off}}})$. Thus, learning is the most efficient, and occurs on the most permissive temporal window, when the postsynaptic neuron has no useful information about x_t (reflected by either the presence or the absence of output spikes), that is, when the postsynaptic neuron is in a complete state of uncertainty. The width of the temporal window for long-term potentiation also depends on r_{on} and r_{off}. It is related to the effective time constant of the postsynaptic neuron, $\tau_{\bar{L}} = \frac{1}{r_{\text{on}}e^{-\bar{L}}+r_{\text{off}}e^{\bar{L}}}$, where $\bar{L}$ is the neutrally stable log-odds for x_t (see the companion letter).

4.4 Temporal Interactions Between Output Spikes. More generally, learning depends on the entire input and output spike train and cannot be reduced to a combination of contribution from pairs of input-output spikes, as it is usually described in the literature. This is due to the nonlinearities in probabilistic computation. Figure 5C shows the contribution of a particular postsynaptic spike (spike 1) to learning as a function of the timing of a second postsynaptic spike (spike 2) when both occur just after a presynaptic spike (spike 0). The contribution of a postsynaptic spike is defined as the difference in the predicted synaptic weight when this spike is present rather than absent. Let us first place ourselves in the optimal situation for learning: the firing rate of the postsynaptic neuron is around $\frac{-\theta_o}{w_o}$. If all input-output spike pairs contributed independently to learning, the contribution of each output spike should be constant regardless of the timing of other spikes. However, we found that the contribution of spike 1 depended critically on whether spike 2 occurred between input spike 0 and spike 1 or after spike 1 (see Figure 5C). In this example spike 2 masks the contribution of spike 1, and there is an apparent competition among input-output spike pairs spike0-spike1 and spike0-spike2 (plain line). If we now consider a case when the output firing rate is 0 (and as a consequence, the neuron belief about x_t is much below the prior, at approximately $\bar{L} = L_o - 2$), the picture is very different. This time spike 1 unmasks spike 2, and the two pairs appear to cooperate rather than compete (dashed line).

Both types of interactions have been reported in different preparations in vitro (Froemke & Dan, 2002; Wang, Gerkin, Nauen, & Bi, 2005). In fact, the presence of apparent competition versus cooperation for successive output spikes depends not only on the output firing rates but also on g_o and the transition rates. Cooperation occurs when the first output spike brings the neuron's belief L_t closer to the prior, leading to a strong influence of the second output spike (very informative in this context). Competition occurs when the first spike brings L_t far above the prior, leading to a weaker effect of the second output spike. Thus, we predict no generic description for interactions between input and output spike pairs in terms of cooperation or competition. Rather, the effect critically depends on the context (in a way that can be predicted by our model).

4.5 LTP or LTD as a Function of Pre- and Postsynaptic Activity. By implementing the online EM, we ensure that the weights converge to the true conditional probabilities (or exactly how informative a synapse is about the hidden state). Thus, we avoid the unbounded growth of the weights usually associated with Hebbian learning rules, without introducing additional competition or weight normalization (Senn & Buchs, 2003). This rule is comparable to a BCM rule (Bienenstock, Cooper, & Munro, 1988) where weights are potentiated or depressed depending on the mean activity of the pre- and postsynaptic neuron.

This is illustrated in Figure 5C. Here, we plot the change in synaptic weight in response to pairing 10 100 ms pulses of presynaptic activity with 10 simultaneous 100 ms pulses of postsynaptic activity, spaced by 1 second. The output firing rate during the pulse of postsynaptic activity is constant (100 spikes per second). The base input firing rate (outside of the pulse) is 20 spikes per seconds. During the pulse of presynaptic activity, the input firing rate is varied from 20 spikes per second (i.e., there is no input pulse) to five (the presynaptic firing rate is five times larger than the base rate during the input pulse, at 100 spikes per second). In addition, we varied the base output firing rate outside the pulse. The results are plotted as a function of the resulting mean belief for $x_t = 1, \bar{L}^{\text{tot}}$. As we can see on Figure 5D, the input weight w_i is potentiated when the presynaptic pulse is strong but also when the mean level of certainty of the postsynaptic neuron $\bar{L}^{\text{tot}}$ is low. The weight is depressed otherwise. The zero contour for the change in weights (i.e., when the coupled pulses result in neither potentiation nor depression of the weight; thick grey line in Figure 5C) corresponds to the case when the strength of the coupling between the input and output spike trains is exactly compatible with the initial synaptic weight before learning, set at $w_i = 0.5$ (i.e., when the right-hand side of equation 4.3 is 0.5).

5 Discussion and Conclusion

We proposed previously that leaky integrate-and-fire neurons act as Bayesian integrators, accumulating evidence about the state of a particular hidden variable. They emit spikes to signal an increase in the probability of this variable. In this letter, we show that single neurons could learn the statistics of this hidden variable from their input, using unsupervised learning inspired from the field of machine learning. The model is self-consistent—the spiking output of one neuron can be used as input and training set for another neuron. Thus, these neurons can be used as building blocks for learning a hierarchy of hidden causes for the sensory input, successive layers of neuron corresponding to successive levels in the hierarchy.

5.1 Synaptic Spike-Time Dependent Plasticity. Online expectation maximization requires neurons to estimate online several sufficient statistics to describe their input (two global statistics and two local statistics per synapse). These statistics can be thought as leaky integration of the synaptic input s_t on a much slower timescale than inference (i.e., synaptic integration to compute L_t). As a result, using bayesian principles, we can predict the changes in synaptic weights and neural dynamics resulting from particular input-output spike patterns, on various timescales, with only one free parameter, the "jump" g_o. Roughly, the weights measure the input-output synchrony. The neural dynamics adapt to account for the probability of the neuron to switch from an inactive to an active state, and vice versa.

The resulting learning rule is local, since it is based on input-triggered averages of the postsynaptic activity. It is comparable to a BCM rule (Bienenstock et al., 1988) since weights are increased or decreased depending on whether pre- and postsynaptic activity is above or below their average value. As a consequence, learning depends on the covariance between pre- and postsynaptic spiking activity.

It has been shown that both long-term potentiation (LTP) and long-term depression (LTD) can be obtained at the same synaptic site by pairing presynaptic pulses of activity with postsynaptic depolarization (Lisman & Spruston, 1986; Artola, Brocher, & Singer, 1990; Dudek & Bear, 1992). The sign of the change in synaptic strength depends on both the presynaptic stimulation rate and the amplitude of postsynaptic depolarization. LTP is observed if the postsynaptic potential is above a particular threshold, and LTD occurs otherwise (Artola et al., 1990). Similarly, LTP is observed if the presynaptic stimulation rate is above a particular threshold, and LTD occurs otherwise (Dudek & Bear, 1992). This is predicted by our model if we consider that the membrane potential is proportional to L_t (see the companion letter). Low presynaptic firing rate and low postsynaptic potential lead to LTD, and LTP is observed in the reversed situation. The presynaptic and postsynaptic activity thresholds for LTP or LTD depend on the synaptic strength (i.e., the synapse is potentiated only if the induced statistical dependency between presynaptic spikes and postsynaptic activation is stronger than predicted by the initial synaptic strength). To our knowledge, this prediction has not been tested experimentally.

Another critical aspect of the Bayesian learning rule is its dependence on precise input and output spike times. Spike-time-dependent plasticity rules have been observed in various brain areas (Markram & Tsodyks, 1996). These experiments have been performed in vitro by inducing input and output spikes through extracellular or intracellular current injection. A presynaptic spike occurring just before a postsynaptic spike induces LTP, while the reverse situation induces LTD. In contrast, our model predicts LTP when pre- and postsynaptic spikes co-occur within a short temporal window, regardless of their temporal order, and LTD when output spikes or input spikes occur in isolation.

Note that plasticity in our model does not truly require postsynaptic spikes, since the online EM algorithm is strictly applied to the synaptic input. In contrast, backpropagating action potential (bAp) is essential for the expression of antisymmetric spike-timing-dependent plasticity (STDP) in vitro (Markram, Lübke, Frotscher, & Sakmann, 1997; Sjöstrom, Turrigiano, & Nelson, 1996), even if they are neither necessary nor sufficient for inducing LTP or LTD (Lisman & Spruston, 1986). Thus, it would seem that the Bayesian learning rule described in this letter might account for synaptic plasticity based on presynaptic spikes and postsynaptic depolarization, but not for bAP-dependent antisymmetric STDP.

However, preliminary results suggest that in a more general context, Bayesian learning also accounts for antisymmetric STDP. We found that LTD when an input spike immediately follows an output spike is required in the presence of feedback connections going from the causes (the tiger) to the consequences (the stripes). Feedback connections are needed to perform full Bayesian inference since the presence of a tiger increases the probability of stripes, not just the other way around. In contrast, the networks considered here have only feedforward connections. A network with feedforward and feedback connections contains many loops. As a result, a neuron needs to "decide" whether a synaptic event is new information or just a result of one of its own spikes reverberated through the network. One way to solve this problem is to predict away the reverberated spikes, an operation that results in depression of synaptic strength if the input spike occurs right after an output spike (and thus is likely to be a reverberated spike). However, this goes beyond the scope of this letter, concentrating on feedforward processing. Learning and inference in recurrent neural networks will be considered in future work.

How well experimental data fit with the model predictions and whether they can account for STDP in vitro or in vivo requires further scrutiny. With the model as it is, we can still note some qualitative similarities between the results of experiments probing in vitro plasticity and nonlinearities observed in our model. Experiments with spike triplets (Froemke & Dan, 2002) and quadruplets (Wang et al., 2005) have shown that synaptic plasticity depends not only on input-output spike pairs, but also on the exact local configuration of the input and output spike in the trains (Yao & Dan, 2005; Froemke & Dan, 2002; Froemke, Tsay, Raad, Long, & Dan, 2006; Wang et al., 2005). This suggest that spike-dependent plasticity rules are not appropriately described in terms of input-output pairs but rather depend, as in our model, on the entire history of input and output spikes.

In particular, competition (Froemke & Dan, 2002; Froemke et al., 2006) between successive input-output spike pairs has been reported. The contribution of input and output spikes to LTP or LTD is diminished by immediately preceding spikes. This can also be the case in our model, as illustrated in Figure 5C (solid line). This result seems to contradict other reports of cooperation between successive spike pairs (Sjöstrom et al., 1996; Wang et al., 2005). For example, in most experiments, LTP or LTD requires the repetition of a sufficient number of spike pairs, above a minimal frequency (10 Hz). Cooperation between pairs is also observed in our model (as illustrated in Figure 5C, dashed line). More generally, we predict that the ability of an input-output spike pair to induce LTP is strong if the log-odds of x_t at the time of the input spike is close to the prior and weak otherwise. A preceding spike could bring the log-odds closer (cooperation) or further (competition) from this prior, depending on the statistics of x_t and the previous history of spikes. In any case, we expect cooperation at long interspike intervals (e.g., LTP requires a minimal frequency of 10 Hz to bring $p(x_t)$ close to the prior) and competition at short interspike intervals (e.g., the first spikes in a burst contributes the most to the potentiation of the synapse).

5.2 Plasticity in Neural Dynamics. Time is an important dimension for dealing with a perpetually changing world, and it is essential to consider this dimension in a spike-based code. The properties of the temporal dynamics of a neuron (such as the time constant for synaptic integration) are as important as its synaptic strength in defining how it will respond to its input. Strikingly, aspects dealing with plasticity of the temporal dynamics of synaptic integration and spike generation are rarely considered as forms of learning. However, these dynamics depend on many parameters, such as position on the dendrite, distributions, and types of receptors, all subject to short-term and long-term activity-dependent changes.

We propose that neurons learn to adapt their own integrative properties to the temporal statistics of their input. This will be reflected, for example, in the temporal constant for synaptic integration (τ_L). They learn the transition rates by estimating the rate at which x_t switches ON and OFF. Changes in the time constant of integration could be implemented by changes in channel densities, structural changes (shape or position on the dendrite of the synapse), selection of particular receptor types (i.e., AMPA versus NMDA receptors), or selection of particular subtypes of neuron. Whether neurons or neural circuits can adapt to the temporal statistics of their input and on which timescale is a question that remains largely to be explored.

5.3 Learning Stability. Another time constant important for the learning performance of the Bayesian neuron is the temporal window for estimating the new parameters, $\tau \approx \frac{1}{\eta}$ (see equation A.12). In fact, while inference corresponds to integrating the synaptic input with a short time time constant τ_L, learning can be considered as a synaptic integration with a much longer time constant τ. τ is a meta-parameter that should also be adjusted to the higher-order temporal statistics of the sensory input. In a word were the parameters of the HMM never to change, τ should be infinitely large to limit the fluctuations of the learned parameters around their true value. However, if the parameters of the HMM change over time, for example, if the stimulus suddenly becomes faster or more informative, a large τ would prevent the neuron from adapting to this new context. Similarly, the neuron can afford a shorter time τ in more informative contexts. This trade-off is described in more detail elsewhere (Mongillo & Deneve, 2006).

5.4 Different Kinds of Spike-Dependent Adaptation. We propose a specific interpretation for spike-dependent adaptation, or, as it is equivalent in the model, the reset in membrane potential after a spike. Spike adaptation ensures that neurons convey new information about the stimulus that has not been conveyed by previous spikes. The amplitude of the jump g_o corresponds specifically to the strength of new synaptic evidence needed to fire spike. For example, $g_o = \log(2)$ implies that a spike is fired whenever the occurrence of new synaptic events multiplies the odds for $x_t = 1$ by 2. Meanwhile, g_o controls the input-output gain, since $\bar{O} \approx \frac{\bar{I}}{g_o}$.

g_o itself might, and should, be adapted to the information received by the neuron. Indeed, neurons have limited dynamical range, while the information contained in their synaptic input, expressed by the mean input $\bar{I}$, is likely to vary several orders of magnitudes with the context. For example, high-contrast stimuli are much more informative than low-contrast stimuli. In a highly informative context, small fluctuations in probability are uninteresting. Thus, one might want to limit the cost of firing spikes with a large g_o. On the other hand, in a low informative context, small fluctuations in probability are potentially meaningful. Small g_o will allow a precise representation of the probability without costing too many spikes. There is now a wide body of evidence that neurons in various sensory systems adapt their response gain to match their own dynamical range with the range of input they receive, thus maximizing information transfer (for a review, see Wainwright, 1999). In the H1 motion-sensitive neuron of the fly, for example, adaptation to the range of stimulus motion is so fast that it is of the order of the performance achievable by an ideal observer (Fairhall, Lewen, Bialek, & de Ruyter van Steveninck, 2001).

In our case, this would correspond to adapting g_o (an adaptation of spike-based adaptation) so that it is proportional to the mean input when the stimulus is present, $\bar{I}_{\text{on}} = \sum_i w_i q^i_{\text{on}} - \theta$. Note, however, that changing g_o also changes the meaning of an output spike. In our framework, this is not a problem since an efferent neuron does not need to know g_o in order to interpret the output spikes. The weights w_i and bias θ can be adjusted online to compensate for changes in the information content of a spike simply by applying the online EM algorithm on a fast timescale. Thus, synaptic short-term plasticity could collaborate with spike adaptation in order to perform the same computation with many fewer spikes in highly informative contexts.

5.5 Limitations and Extensions of This Approach. This work proposes a simple neural implementation of Bayesian inference and learning in corresponding hierarchical generative models. However, since we considered networks with only pairwise interactions (i.e., synaptic transfer) between the variables, we were strongly limited in which generative models we could implement. One obvious limitation of our approach is that it uses only binary variables. We are currently exploring straightforward extension of the approach to multistate variables and continuous variables.

Another strong limitation is that we did not consider cases when variables can have several potential causes. This is due to the well-known problem of explaining away in hierarchical generative models: alternative causes to the same event must compete for the evidence they receive from this event (Frey, 1998). A related issue is that we did not implement feedback processing, that is, propagation of evidence from the causes to the consequences. We are currently developing more complex recurrent spiking networks that implement both explaining away and feedback propagation of evidence. These spiking networks can perform only approximate inference.

Finally, this approach is limited to very simple temporal dynamics where only objects at the top of the hierarchy can have an independent history. This assumption will not be valid for more complex dynamics with several interacting dynamical processes, such as the motion of an object, described by position, speed, and acceleration. One solution could be to group variables, for example, to use one neuron for each combination of position, speed, and acceleration. This is clearly a costly solution in terms of number of neurons and connections, but cortical neurons do seem to respond preferentially to particular combinations of statistically or dynamically related states (Poggio, 1990).

Appendix: Online Expectation-Maximization Algorithm

Learning consists of maximizing the likelihood of the observable sequence $\mathbf{s}_{0\to t}$ with respect to the model parameters—the transition (r_{on}, r_{off}) and the emission rates (q_{on}, q_{off}).

In order to do this, we use the expectation-maximization (EM) algorithm, an iterative optimization technique for maximum-likelihood estimation when the data set is incomplete or has missing values. In our case, the missing values, as they are not directly observable, are the sequence of hidden states $x_{0\to T}$ that produced the observable sequence $\mathbf{s}_{0\to t}$.

For simulations, and also for some parts of the theory, it is convenient to formulate the learning algorithm in discrete time with step size dt. The continuous time limit, where necessary, is obtained by taking $dt \to 0$. In the case of HMM, the EM algorithm reduces to a reestimation of the parameters: the Baum-Welch reestimation formulas (see Rabiner, 1989).

For example, an estimate of the transition rate from the ON to OFF state, r_{off}, is obtained as the expected number of transitions ON$\to$OFF over a given time interval T, $N_{on\to off}(T)$, divided by the time spent in the ON state over the same time interval, $\tau_{\text{on}}(T)$, and the time step dt:

$$\begin{aligned} N_{on\to off}(T) &= \sum_{t=1}^{T}\sum_{i=0}^{1}\sum_{j=0}^{1}[i\cdot(1-j)]\cdot P(x_{t-dt}=i, x_t=j|\mathbf{s}_{0\to t}) \\ &\equiv E\left(\sum_{t=1}^{T} x_{t-dt}\cdot(1-x_t)|s_{0\to T}\right) \end{aligned} \tag{A.1}$$

$$\tau_{\text{on}}(T) = \sum_{t=0}^{T}\sum_{i=0}^{1} i\cdot P(x_t=i|\mathbf{s}_{0\to t}) \equiv E\left(\sum_{t=0}^{T} x_t|s_{0\to T}\right) \tag{A.2}$$

$$r_{\text{off}} = \frac{1}{dt}\cdot\frac{N_{\text{on}\to\text{off}}(T)}{\tau_{\text{on}}(T)}. \tag{A.3}$$

Equation A.3 computes a reestimate of r_{off} as a function of the current parameters' values, which are used in computing the left-hand side of equation A.3 via equations A.1 and A.2. Analogous reestimation formulas hold for the remaining parameters:

$$r_{\text{on}} = \frac{1}{dt}\cdot\frac{N_{\text{on}\to\text{off}}(T)}{T-\tau_{\text{on}}(T)} \tag{A.4}$$

$$q_{\text{on}} = \frac{1}{dt}\cdot\frac{N_{\text{on}}^{(sp)}(T)}{\tau_{\text{on}}(T)} \tag{A.5}$$

$$q_{\text{off}} = \frac{1}{dt}\cdot\frac{N^{(sp)}(T)-N_{\text{on}}^{(sp)}(T)}{T-\tau_{\text{on}}(T)}, \tag{A.6}$$

where $N_{\text{on}}^{(sp)}(T)$ is the expected number of spikes during time interval T, fired while in the ON state, and $N^{(sp)}(T)$ is the total number of spikes in the same interval. Note that in equation A.4, we take $N_{off\to on}(T) = N_{on\to off}(T)$ in the limit of large T. It can be shown that any function of the type

$$\phi(T) = E\left(\sum_{t=1}^{T} h(x_{t-dt}, x_t, s_t)|s_{0\to T}\right) \tag{A.7}$$

can be computed recursively with T in the following way. Let $\phi_j(T)$ be

$$\phi_j(T) = E\left(\sum_{t=1}^{T} h(x_{t-dt}, x_t, s_t)|x_T = j, s_{0\to T}\right) \cdot P(x_T = j|s_{0\to T}). \tag{A.8}$$

Clearly, $\sum_j \phi_j(T) = \phi(T)$. From probability calculus, the following recurrence relation for the ϕ_j's holds true:

$$\begin{aligned}\phi_j(T) = \sum_i m_{ij}(s_T, s_{0\to T-dt}) \cdot [\phi_i(T-dt)\\ + h(i, j, s_T) \cdot P(x_{T-dt} = i|s_{0\to T-dt})],\end{aligned} \tag{A.9}$$

where

$$m_{ij}(s_T, s_{0\to T-dt}) = \frac{P(s_T|x_T = j) \cdot P(x_T = j|x_{T-dt} = i)}{P(s_T|s_{0\to T-dt})}. \tag{A.10}$$

The expected number of transitions ON→OFF, $N_{on\to off}(T)$, the expected number of spikes while in the ON state, $N_{\text{on}}^{(sp)}(T)$, and the expected time spent in the ON state, $\tau_{\text{on}}(T)$, are in the form of equation A.7, for suitably chosen $h(x_{t-dt}, x_t, s_t)$, that is,

$$\begin{cases} h(x_{t-dt}, x_t, s_t) = \delta(x_{t-dt} - 1)\cdot\delta(x_t) & \to \quad N_{\text{on}\to\text{off}} \\ h(x_{t-dt}, x_t, s_t) = \delta(x_t - 1)\cdot\delta(s_t - 1) & \to \quad N_{\text{on}}^{(sp)} \\ h(x_{t-dt}, x_t, s_t) = \delta(x_t - 1) & \to \quad \tau_{\text{on}}, \end{cases} \tag{A.11}$$

where $\delta(\cdot)$ is the Kronecker delta. Thus, the statistics required for reestimating the model parameters via equations A.3 to A.6 can be computed online via equations A.9 and A.10. In order to deal with infinite observable sequences, we compute the online averages equation A.7 on a sliding time window according to

$$\phi(T) = E\left(\sum_{t=1}^{T} e^{\eta\cdot(T-t)} h(x_{t-dt}, x_t, s_t) | s_{0\to T}\right), \tag{A.12}$$

where η is a suitably chosen decay factor (i.e., it should be much longer than the fastest timescale in the system). Hereafter, $e^{\eta\cdot(T-t)}$ is referred to as a forgetting function. For constant model parameters, $\phi(T)$ in equation A.12 is, to a good approximation, the expectation value of $\sum_t h(x_{t-dt}, x_t, s_t)$ over a time window of length $\tau \sim 1/\eta$ ending at time step T. Then we use the current sufficient statistics to obtain a new estimate of the parameters,

$$\theta(T) = H(\phi(T)), \tag{A.13}$$

where $\theta(T)$ indicates the model parameters at time step T and $H(\cdot)$ is a compact notation for the map, equations A.3 to A.6. The new model parameters are then used to update the online sufficient statistics in the next time step according to equations A.9 and A.10. To summarize, the online learning algorithm works in the following way. Starting with an initial guess for the parameters, at each time step, the online sufficient statistics, equation A.12, are updated depending on their value at the previous time step, $\phi(T-dt)$, the actual input, s_T, and on the parameters estimate at the previous time step, $\theta(T-dt)$,

$$\phi(T) = F(\phi(T-dt), s_T, \theta(T-dt)), \tag{A.14}$$

where $F(\cdot)$ is a compact notation for the map, equations A.9 and A.10. Then the current sufficient statistics is used to obtain a new estimate at the current time step via equation A.13. The learning algorithm does not require any extra storage of incoming data; it is completely online and updates the model parameters as soon as new data are available. Furthermore, it can be shown that for suitably chosen forgetting functions, the online algorithm is equivalent to a stochastic approximation for obtaining the maximum likelihood estimate for the parameters and, in this sense, it is equivalent to the batch EM algorithm (Mongillo & Deneve, 2006).

Acknowledgments

This work was supported by the European consortium BACS FP6-IST-027140 and a Marie Curie Team of Excellence Fellowship BIND MEXT-CT-2005-024831. We thank the members of the Gatsby unit, in particular Peter Dayan, for fruitful discussions.

References

Artola, A., Brocher, S., & Singer, W. (1990). Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex. *Nature, 347*(6288), 69–72.

Barlow, H. (2001). Redundancy reduction revisited. *Network, 12*(3), 241–253.

Bell, A., & Sejnowski, T. (1995). A non-linear information maximization algorithm that performs blind separation. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems, 7*. San Mateo, CA: Morgan-Kaufmann.

Bienenstock, E., Cooper, L., & Munro, P. (1988). Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience, 2*(1), 32–48.

Bishop, C. (1995). *Neural networks for pattern recognition*. Oxford: Clarendon Press.

Bliss, T. V. P., & Collingridge, G. L. (1993). A synaptic model of memory: Long-term potentiation in the hippocampus. *Nature, 31*(39), 1715–1749.

Dudek, S., & Bear, M. (1992). Homosynaptic long-term depression in area CA1 of hippocampus and effects of N-methyl-D-aspartate receptor blockade SM Dudek and MF Bear. *Proc. Natl. Acad. Sci. U.S.A., 89*, 4363–4367.

Fairhall, A., Lewen, G., Bialek, W., & de Ruyter van Steveninck, R. (2001). Efficiency and ambiguity in an adaptive neural code. *Nature, 412*(6849), 776–777.

Frey, B. (1998). *Graphical models for machine learning and digital communication*. Cambridge, MA: MIT Press.

Froemke, R., & Dan, Y. (2002). Spike-timing-dependent synaptic modification induced by natural spike trains. *Nature, 416*(6879), 433–438.

Froemke, R., Tsay, I., Raad, M., Long, J., & Dan, Y. (2006). Contribution of individual spikes in burst-induced long term modification. *Journal of Neurophysiology, 95*(3), 1620–1629.

Hebb, D. (1949). *Organization of behavior: A neuropsychological theory*. New York: Wiley.

Hebb, D. (1984). *Self-organization and associative memory*. New York: Springer.

Jordan, M. (1974). *Learning in graphical models*. Cambridge, MA: MIT Press.

Lee, T., & Mumford, D. (2003). Hierarchical Bayesian inference in the visual cortex. *J. Opt. Soc. Am. A Opt. Image. Sci. Vis., 20*(7), 1434–1448.

Lisman, J., & Spruston, N. (1986). From basic neural network principles to neural architecture: Emergence of orientation columns. *Nature Neuroscience, 83*, 839–841.

Markram, H., Lübke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science, 275*, 213–215.

Markram, H., & Tsodyks, M. (1996). Redistribution of synaptic efficacy between neocortical pyramidal neurons. *Nature, 382*, 807–819.

Mongillo, M., & Deneve, S. (2006). *Online line expectation-maximization in hidden Markov models.* Unpublished manuscript submitted for publication.

Poggio, T. (1990). A theory of how the brain might work. *Cold Spring Harbor Symposium on Quantitative Biology, 55*, 899–910.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77*(2), 4–16.

Senn, W., & Buchs, N. (2003). Spike-based synaptic plasticity and the emergence of direction selective simple cells: Mathematical analysis. *Journal of Computational Neuroscience, 14*(2), 119–138.

Sjöstrom, P., Turrigiano, G., & Nelson, S. (1996). Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neural Computation, 8*(3), 511–530.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory, 13*(2), 260–267.

Wainwright, M. (1999). Visual adaptation as optimal information transmission. *Vision Research, 39*(23), 3960–3974.

Wang, H., Gerkin, R., Nauen, D., & Bi, G. (2005). Coactivation and timing-dependent integration of synaptic potentiation and depression. *Nature Neuroscience, 8*(2), 187–193.

Weiss, Y., & Freeman, W. (2001). Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation, 13*, 2173–2200.

Yao, H., & Dan, Y. (2005). Synaptic learning rules, cortical circuits, and visual function. *Neuroscientist, 11*(3), 206–216.

Received June 12, 2006; accepted March 19, 2007.