Belief Propagation in Networks of Spiking Neurons

Andreas Steimer
asteimer@ini.phys.ethz.ch
Institute of Neuroinformatics, University of Zürich, and ETH Zürich,
Zürich, 8057 Switzerland

Wolfgang Maass
maass@igi.tugraz.at
Institute for Theoretical Computer Science, Technische Universität Graz,
A-8010 Graz, Austria

Rodney Douglas
rjd@ini.phys.ethz.ch
Institute of Neuroinformatics, University of Zürich, and ETH Zürich,
Zürich, 8057 Switzerland

From a theoretical point of view, statistical inference is an attractive model of brain operation. However, it is unclear how to implement these inferential processes in neuronal networks. We offer a solution to this problem by showing in detailed simulations how the belief propagation algorithm on a factor graph can be embedded in a network of spiking neurons. We use pools of spiking neurons as the function nodes of the factor graph. Each pool gathers "messages" in the form of population activities from its input nodes and combines them through its network dynamics. Each of the various output messages to be transmitted over the edges of the graph is computed by a group of readout neurons that feed in their respective destination pools. We use this approach to implement two examples of factor graphs. The first example, drawn from coding theory, models the transmission of signals through an unreliable channel and demonstrates the principles and generality of our network approach. The second, more applied example is of a psychophysical mechanism in which visual cues are used to resolve hypotheses about the interpretation of an object's shape and illumination. These two examples, and also a statistical analysis, demonstrate good agreement between the performance of our networks and the direct numerical evaluation of belief propagation.

1 Introduction

1.1 Graphical Models and the Belief Propagation Algorithm. Many computational problems faced by nervous systems involve inferring unknown information from incomplete sensory data combined with some prior knowledge, and so can be considered a form of statistical inference (Kersten, Mamassian, & Yuille, 2004; Ernst & Banks, 2002; Körding & Wolpert, 2004; von Helmholtz, 1867; Kersten & Yuille, 2003). These problems usually comprise a set of observed variables whose state is known and a second set of hidden variables whose states must be estimated probabilistically from the known variables by computing their (conditional) marginal probability distributions. For many problems, there are dependencies between only some of the variables; hence, graphs are an elegant way of visualizing and computing on these dependencies. For example, in graphical models like factor graphs (Kschischang, Frey, & Löliger, 2001; Bishop, 2006), a factor node expresses a relation between the variables the node is connected to and provides a nonnegative scalar value for each combination of states of these variables. Multiplying the values of all nodes together provides a factorized representation of the probability of any joint state. The structure of the graph can be used as a starting point for calculating quantities like the maximum a posteriori (MAP) estimate or the marginal probabilities associated with each unobserved variable (Kschischang et al., 2001; Löliger, 2004; Löliger et al., 2007; Bishop, 2006). By contrast, computing marginal probabilities by straightforward summation is not feasible for nervous systems. First, it is inefficient because the number of operations scales exponentially with the number of variables, and second, it requires a summation mechanism that has global access to all states of all variables, a requirement that is incompatible with biology's physical constraint of local information processing.

The belief propagation (BP) algorithm (Kschischang et al., 2001; Löliger, 2004; Bishop, 2006) avoids these problems, and so is a plausible mechanism for brain computation. The algorithm employs computational units (the nodes of a graphical model) that communicate by distributing messages exclusively to their neighbors in the graph. An outgoing message from a source node can be regarded as an estimate of the state of a particular variable. This estimate depends on local information given by the input that the node receives from its neighbors. After the message-passing dynamics have converged, a node multiplies all (locally available) messages that depend on the same variable and normalizes across its states to yield the variable's (possibly approximate) marginal probability distribution. The advantage of this message-passing scheme compared to straightforward summation lies in the decomposition of a large inference problem containing exponentially many terms to compute, into several small subparts, each of which can be solved locally by a node of the graph in a parallel manner (see section 2).

If the sum-product rule (Kschischang et al., 2001; Löliger, 2004) is used to compute an outgoing message from the incoming ones, the BP algorithm is guaranteed to converge to the exact marginal distributions for tree-structured graphical models, but not necessarily for general graphs containing cycles (Yedidia, Freeman, & Weiss, 2005; Löliger, 2004).

However, even in these cyclical cases, the algorithm often performs surprisingly well (Yedidia et al., 2005).

Several recent studies have considered how single neurons or their networks could implement BP (Rao, 2004; Ott & Stoop, 2006) or of how they could perform probabilistic computations in general (Yu & Dayan, 2005; Deneve, Latham, & Pouget, 2001; Deneve, 2007; Ma, Beck, Latham, & Pouget, 2006). However, the neuronal level models offered so far are limited in that they are confined to a restricted set of graphical models (Deneve, 2007), use restricted inference procedures (Ma et al., 2006), or depend on rough approximations (Rao, 2004).

In this study, we focus instead on a neuronal network approach in which BP dynamics play out on an abstract graphical structure, a Forney-factor-graph (FFG; Löliger, 2004), that is embedded in a network of spiking neurons. Each node in the graph represents a single factor of a factorized joint probability distribution and, at the neuronal level, can be associated with a collection of liquid state machines (LSMs; Maass, Natschläger, & Markram, 2002). As in a typical LSM, our graphical node is composed of two kinds of neuronal populations: a pool of recurrently connected neurons whose rich dynamics implicitly perform nonlinear transformations of the input (liquid pool) and additional pools of neurons that compose particular output functions as linear combinations of these transformations (readout pools). The output functions constitute the output messages transmitted across the edges of the abstract graph (see Figure 1). At the implementation level, these messages are encoded in the spike patterns traversing the axons of the neurons in the readout pools. All messages arriving at a node are fed into the single liquid pool, whose nonlinear projections of the input are crucial for implementing the multiplicative message update rule (sum-product rule) of the BP algorithm. Each of the various pools of readout units within a node computes the output message associated with a particular edge of the graph. Both the liquid and the readout pools consist of leaky integrate-and-fire units.

We demonstrate the concept illustrated in Figure 1 by two different examples of inference problems. The first example models a problem from coding theory involving signal transmission through a noisy channel (see Figure 3). We choose this nonbiological problem because it demonstrates well the principles and generality of our approach. In the second example, we model a psychophysical mechanism where visual cues are used to resolve hypotheses about the interpretation of an object's shape and illumination in a visual scene (see Figure 4).

2 Methods

2.1 Specific Version of the Belief Propagation Algorithm. Depending on the context, the belief propagation (BP) algorithm can be formulated in different ways, and various graphical models with different message

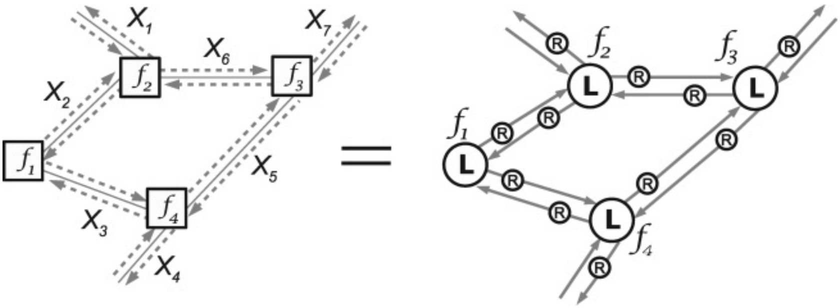


Figure 1: Neural implementation of belief propagation (BP) on a network of recurrently connected liquid state machines. (Left) Symbolic illustration of BP in a Forney factor graph (Löliger, 2004) (nodes represent factors $f_1, \ldots, f_4$ and edges the variables $X_1, \ldots, X_7$). The product $P(X_1, \ldots, X_7) \propto f_1(X_2, X_3) \cdot f_2(X_1, X_2, X_6) \cdot f_3(X_5, X_6, X_7) \cdot f_4(X_3, X_4, X_5)$ then provides a factorized representation of the joint probability distribution $P(X_1, \ldots, X_7)$. The BP messages exchanged between neighboring nodes are indicated by arrows along the edge connecting two neighbors. (Right) Schematic showing the principles of the neural network implementation of BP in the graph at left. L and R, respectively, represent liquid and readout populations of neurons in a liquid state machine. Each message (arrows) is computed by a combination of a pool of liquid and readout neurons. The liquid pool represents the factor of the node sending the message. Messages are encoded by the population rate of the corresponding readout population and are injected into the target liquid pool through synapses.

update rules have been described (Pearl, 1988; Kschischang et al., 2001; Löliger, 2004; Ott & Stoop, 2006). We use the Forney factor graph (FFG) model, in which the nodes of the graph correspond to factors and the edges to variables (see Figure 1). We use this form because the formulation of BP is simplest in these graphs. For example, in contrast to BP in ordinary (bipartite) factor graphs, FFGs have only a single type of message (from factor node to factor node), which is a canonical strategy and more likely to be implemented by the cortex.

In a FFG, a message $m_{f_1 \to f_2}(X)$ sent from a node f_1 to a node f_2 along the variable X provides a biased estimate about the states of X by summarizing the subgraph $\mathcal{G}_1$ behind (with respect to the direction of the message) f_1. That means that in a marginalization task, $m_{f_1 \to f_2}(X)$ contains all summations associated with variables in $\mathcal{G}_1$ (Löliger, 2004). Since (in a tree-structured graph) these summations do not depend on the variables in $\mathcal{G}_2$, the subgraph in front of f_2, they need not be repeated for each (joint) state of the variables in $\mathcal{G}_2$. This property is a consequence of the distributive law and the reason for the much greater efficiency of BP compared to marginalization by direct summation.

The variables in an FFG can be binary, of arbitrary base, or even continuous. For simplicity, in this work, we have focused on the binary type, although in principle our approach is also applicable to variables of arbitrary base. When normalized, a message of a binary variable can be a single scalar value that can be interpreted as the estimated probability of the variable taking the value 1. Thus, normalization also encodes implicitly the probability for the value 0. These estimates are based on information locally available to the node sending a message (incoming messages).

In the computational science domain, the execution time of the BP algorithm is necessarily discrete, allowing different clocked message-passing schedules or schemes. A schedule determines when a message must be computed and to which neighbor it must be transmitted. One of the simplest of these schemes is the flooding schedule (Kschischang & Frey, 1998; Kschischang et al., 2001). There, every message in the graph is computed and transmitted at every time step. This strategy has the advantage that the sum-product rule can be easily formulated as a system of first-order, nonlinearly coupled differential equations in continuous time and therefore is well suited to a neural implementation.

Furthermore, in practice, it is often helpful for the algorithm to converge if the messages are only partially updated at each time step (Yedidia et al., 2005), and the differential version of the flooding schedule can be regarded as the extreme case of an infinitesimal update dm in time step dt. It belongs to the class of time-invariant filters with fading memory, and so can be approximated by a LSM (Maass et al., 2002). In general, the differential version of the flooding schedule for binary variables in a FFG is given by

$$\begin{aligned}&\tau \dot{m}_{i\to j}(t) + m_{i\to j}(t)\\ &\quad = \frac{1}{Z(t)} \sum_{X\in\{0,1\}^n} f_i(x_j = 1, X) \prod_{k\in\mathcal{N}(i)/j} p_{k\to i}(x_k, t-D), \qquad (2.1)\end{aligned}$$

where $m_{i\to j}(t)$ is the (scalar) message passed from node i to node j at time t; τ, an arbitrary time constant determining the overall speed of the computation dynamics; $\mathcal{N}(i)/j$, indicating the set of neighbors of node i besides j; n, the cardinality of $\mathcal{N}(i)/j$; $X = (x_k)_{k\in\mathcal{N}(i)/j}$, the vector of binary variables x_k, each of which links nodes i and k; f_i, the factor associated with i; D, a fixed transmission delay (synaptic delay); $p_{k\to i}(x_k, t) = \{m_{k\to i}(t),\ \text{if } x_k = 1; 1 - m_{k\to i}(t), \text{otherwise}\}$; and $Z(t)$, a normalization term ensuring that $0 \le m_{i\to j}(t) \le 1$. $Z(t)$ therefore depends on the $f_i(x_j = 1, X)$ as well as on the $f_i(x_j = 0, X)$.

2.2 Implementation of the Liquid Pools. Our spiking neuron simulations were implemented using the software package CSIM (available online at the Institute for Theoretical Computer Science/TU-Graz, http://www.lsm.tugraz.at). The organization of our liquid pools followed closely that of Maass (Maass et al., 2002). We used leaky integrate-and-fire (LIF) neurons (20% inhibitory) located on the integer points of cuboid lattices with a distance-dependent gaussian connection probability. A synaptic connection between two neurons with position vectors **a** and **b** was established with probability $p = C \cdot \exp -\frac{\|(\mathbf{a}-\mathbf{b})\|^2}{\lambda^2}$. We used only current injecting synapses, which were modeled as low-pass filters of the input spike trains. That is, after a synaptic delay, each presynaptic spike generated an instantaneous increase of the postsynaptic current that was followed by an exponential decay (see the supplementary material for parameter values).[1]

Spike inputs (SI) to a liquid pool were provided either externally by a population of dummy neurons (used to model prior knowledge or observed variables) or by a readout pool (R) consisting of a population of spiking LIF neurons (used for messages computed within the network; see Figures 3–5). Input neurons of both types connected to neurons of the postsynaptic pool with a probability that was independent of distance.

For the results presented in this letter, neurons in liquid pools did not receive any diffusive noise current (in contrast to neurons belonging to readout populations; see section 3). However, the performance is robust even in the presence of noise in the liquid pool (see supplementary material).

2.3 Neural Encoding of the Messages and Generation of Training Inputs. We have used the population activity/rate (Gerstner & Kistler, 2002) of the readout neurons as a method for encoding the time series of the messages $m_{i\to j}(t)$ in equation 2.1 with spikes. Therefore, the spiking neurons of the readout pools (see Figure 1) should encode messages (probability values) using a population rate code that depends linearly on the encoded scalar message value (probability values 0/1 corresponded to a population rate of 0/90 Hz, respectively; see section 3).

The problem, then, is how to arrange for the readout neurons to report the correct output message in response to the current state of the liquid, which depends on the time series of the input messages. In overview, our solution is to drive the liquid with a broad range of possible input time series (spike trains) and train the readout pools to generate their corresponding output messages. We expect that when suitably trained, these pools will be able to generalize their learned behavior to generate appropriate output messages in response to novel input messages injected into the liquid.

If a general filter of the kind in equation 2.1 is to be implemented by a LSM, the training examples must cover a wide range of possible input time series $m_{k\to i}(t)$, because it is not known in advance what input time series $m_{k\to i}(t)$ liquid pool i will be exposed to once trained. Therefore, training examples were generated as follows: low-pass-filtered white noise (an Ornstein-Uhlenbeck process with reflective bounds) was used to generate random population rate curves, each coding for a possible message signal

[1] Supplementary materials are available online at http://www.mitpressjournals.org/doi/suppl/10.1162/neco.2009.08-08-837.

$m_{k\to i}(t)$. These patterns represented the instantaneous firing rate underlying an inhomogeneous poisson spike process. Since each rate curve determines a population message signal, the same curve determines the instantaneous rate of each neuron in an input population. The training input spike trains had a duration of 120 to 300 s in simulated biological time.

3 Characterizing the Population Code of the Readout Pools

In order to generate the spiking readout messages of our choice, we first had to determine the relationship between the total mean input current (I) provided by the synapses from the liquid neurons to the readout neurons and the population rate (R) of the latter. This step was necessary to ensure consistency between the encoding of the (training) input messages and the readout response, which, after training, provided the input to a neighboring liquid pool. We obtained this relationship empirically by conducting several simulations with independent samples of networks of the following type.

All $N = 343$ neurons belonging to the same readout population were mutually unconnected, and their threshold values were drawn independently from a uniform distribution. Each neuron received a diffusive (white) noise current; however, all of them received the same temporally varying mean input current I (see Silberberg, Bethge, Markram, Pawelzik, & Tsodyks, 2004; Brunel, Chance, Fourcaud, & Abbott, 2001) for similar networks, but without distributed threshold values).

We performed 20 trials, in each of which a random instance of such a network was created. A random mean current trace $I(t)$ was then injected into all the neurons. This current, together with additive gaussian noise, evoked Poissonian spike trains of varying instantaneous firing rates in the neurons. All neurons received the same mean current. However, their individual white noise processes were independent. The corresponding population rate $R(t)$ was the result of the combined firing of all neurons and computed as follows. The spike trains of all neurons in a given trial were merged into a single one, and $R(t) = \frac{1}{N}\frac{dC}{dt}$ was determined, where $C(t)$ is the (smoothed) total number of spikes fired until time t by the whole population. The $I(t)$ and $R(t)$ data of all trials were lumped together and the $I(R)$ relationship determined by polynomial fitting. This relationship was then used to determine the desired target current during all supervised learning procedures. We used a linear dependency between R and the message $m \in [0, 1]$ encoded by it: $R = 90\text{ Hz} \times m$.

After learning, the current I is provided by the sum of (trained) postsynaptic currents (PSCs) injected by the neurons of the presynaptic liquid pool. We found that the distributed firing thresholds in the readout populations have an important role in that they permit the population responses to depend mainly on the actual I and less on past values of this current (M. Bethge personal communication to A. Steimer, May 2007). The goal was to map instantaneously I together with the noise onto a corresponding population rate without introducing dependencies on the distant past, which the liquid pool is unable to provide because of its fading memory (Maass et al., 2002).

3.1 Training the Readout Pools. The response spikes of the liquid neurons were recorded for each training input message and used to compute the resultant synaptic current injected into the neurons of the readout pool.

The desired synaptic current could be derived from the training input message as follows. First, the desired population rate to instantiate the required output message can be calculated from equation 2.1 and the linear dependency between encoded message (probability) $m_{i\to j}(t)$ and population rate. Then the desired input current can be derived from the empirically measured I(R) relationship. The I(R) relationship ensures that this current (together with noise) will result in the correct population rate encoding of the output message $m_{i\to j}(t)$.

For computational simplicity, we used linear regression to adjust the synaptic weights toward the desired input current; however, in future work, this task is expected to be solved within the framework of reward-modulated STDP learning (Legenstein, Pecevski, & Maass, 2008). In our simulations, the synaptic weights were the same for all neurons in the same readout population. Consequently, after successful learning, all the readout neurons received the same mean synaptic current through the synapses from their presynaptic liquid pool.

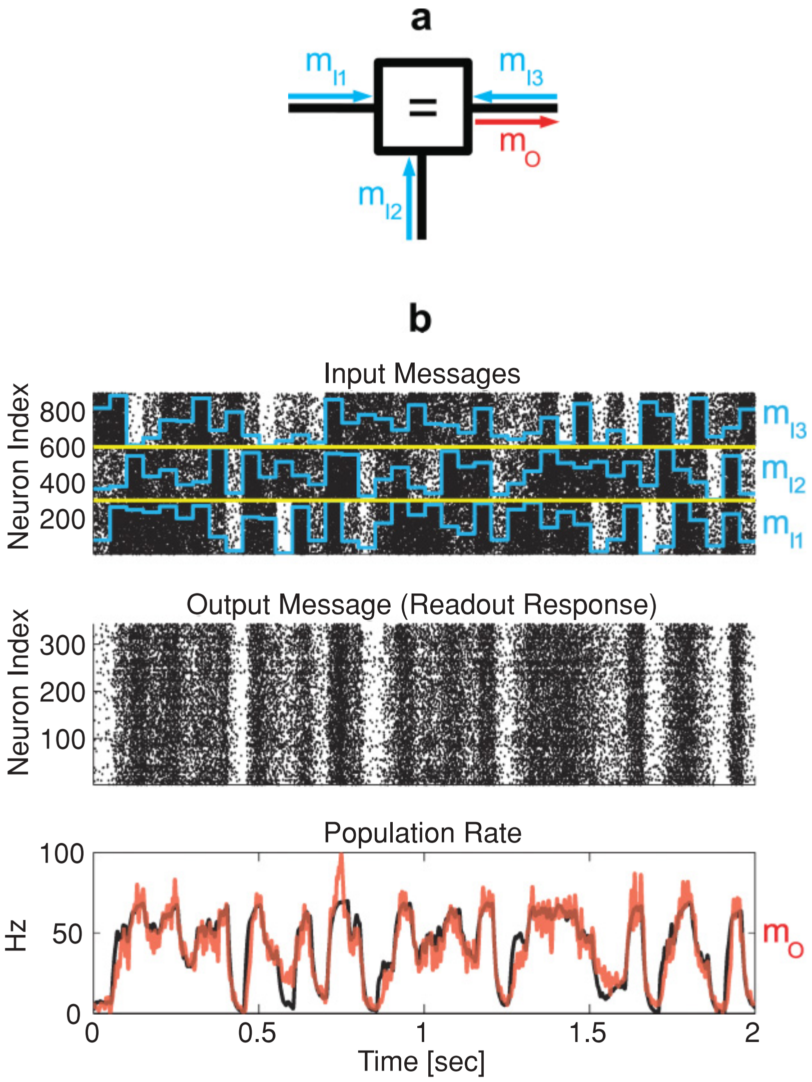
All readout populations were trained separately and then connected to their presynaptic liquid pool.

4 Results

4.1 Evaluation of the Model in a General Inference Problem. Our first task was to evaluate the ability of a single readout population to approximate the dynamics determined by the sum-product rule in equation 2.1. Figure 2 shows the response to test input signals of a single, trained readout population in an isolated factor node. The =-factor defines an equality constraint node as given in Löliger (2004). Since in a FFG, a variable (edge) can be connected to at most two factors, this specific factor has been constructed for the purpose of cloning a variable. Connected to the variables X, Y, Z, it is defined through $f_=(X, Y, Z) := \delta(X - Y)\delta(Y - Z)$ ($\delta(W) = \{1 \text{ if } W = 0; 0 \text{ otherwise}\}$, denoting the (discrete) Kronecker delta function, thereby enforcing equality among X, Y and Z).

When inserted into equation 2.1 the =-factor leads to the following expression for the message m_O in Figure 2:

$$\tau \dot{m}_O(t) + m_O(t) = \frac{m_{I1}(t-D)m_{I2}(t-D)}{m_{I1}(t-D)m_{I2}(t-D) + (1 - m_{I1}(t-D))(1 - m_{I2}(t-D))}. \tag{4.1}$$

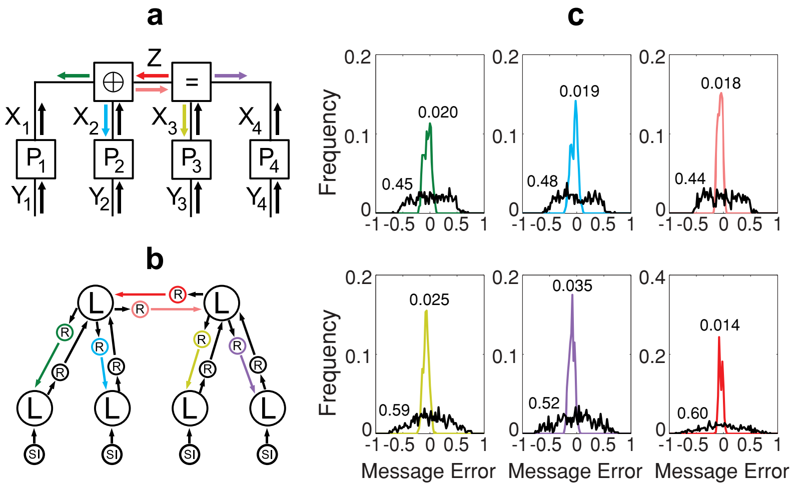


In Figure 2b, the test input time series was drawn from a completely different distribution to that of the training phase. Clearly the readout is able to generalize its behavior to the novel situation, and its population rate is highly correlated with the target output. This ability to train robust individual message outputs provides a foundation for constructing more elaborate factor graphs with interconnected liquid and readout pools, as shown in Figures 3b and 4d.

We first examined the ability of a generic neural network to perform general inference. Here it is the principle rather than the particular (e.g., neurophysiological) application that is at issue, and so we chose to simulate a FFG representing a signal transmission problem that has been described in the literature as an example of standard discrete BP (Löliger, 2004; see Figure 3). The problem is as follows. Suppose each bit X_i of a binary code word $(X_1, \ldots, X_4) \in \{(0, 0, 0, 0), (0, 1, 1, 1), (1, 0, 1, 1), (1, 1, 0, 0)\}$ is fed into an unreliable signal transmission channel, which inverts the bits with a certain probability. Given the observation of all bits at the end of the channel $(Y_1, \ldots, Y_4)$, we wish to know the marginals $P(X_i \mid Y_1, \ldots, Y_4)$.

The model is expressed as the FFG of Figure 3a. The FFG represents a factorization of the a posteriori probability $P(X_1, X_2, X_3, X_4, Z \mid Y_1, Y_2, Y_3, Y_4)$, where Z is an auxiliary variable, X_i the value of bit i at the beginning, and Y_i the observed value of this bit at the end of the channel. The product of the $\oplus$- and $=$-factors connected via Z models membership of a given binary sequence $(X_1, \ldots, X_4)$ to the set of code words above. It can be seen from those code words that the $\oplus$-node implements the parity check function $f_\oplus(X_1, X_2, Z) = \delta(X_1 \oplus X_2 \oplus Z)$ ($\oplus$ denotes the exclusive-or function), and the equality constraint node enforces equality among the variables Z, X_3, X_4. The product of the two factors therefore ensures that for valid code words, both X_3 and X_4 are the parity bit of X_1 and X_2. The empty nodes model the unreliability of the bit-wise signal transmission, that is, they are defined by the conditional probabilities $P_i(Y_i \mid X_i)$. These probabilities are equal to 0.8 for $X_i = Y_i$ and 0.2 otherwise. The product of all factors yields the a posteriori probability $P(X_1, \ldots, X_4, Z \mid Y_1, \ldots, Y_4) \propto f_\oplus(X_1, X_2, Z) f_=(X_3, X_4, Z) \prod_{i=1}^{4} P_i(Y_i \mid X_i)$. By applying message passing

Figure 2: Performance of the computation of a single message in an isolated factor. (a) Schematic of the simulated factor together with the corresponding messages $\{m_{I1}(t), m_{I2}(t), m_{I3}(t), m_O(t)\}$. The factor defines an equality constraint node as defined in Löliger (2004). (b) First row: Example spike trains of the three input populations coding for the messages m_{I1}, m_{I2}, m_{I3}, respectively, with 343 neurons coding each message. Spikes are depicted as single dots. The cyan lines schematically indicate the instantaneous firing rates used to create these input spike trains. Every 50 ms, the rates have been chosen independently for each message according to a uniform distribution. The maximum and minimum rates were 90/0 Hz, representing the probabilities $m_{Ik} = 1/m_{Ik} = 0, k \in \{1, 2, 3\}$, respectively. Second row: Response spike trains of a trained readout population representing the output message $m_O(t)$. Third row: black line: target curve showing the population rate corresponding to the ideal message $m_O(t)$ solving equation 4.1. Red line: actual population rate calculated out of the spike trains in the second row. The encoded probability signal is a linearly scaled version of this rate curve. The correlation coefficient between the red and the black curve is 0.91.



d

Message	$m_{\oplus\to P1}$	$m_{\oplus\to P2}$	$m_{\oplus\to =}$	$m_{=\to P3}$	$m_{=\to P4}$	$m_{=\to\oplus}$
CC	0.52	0.53	0.59	0.93	0.92	0.98

Figure 3: (a) An unreliable binary channel model (Löliger, 2004) (see text for details). (b) Schematic of the neural implementation of the FFG in *a*, using interacting pools of neurons (circles). (c) Error histograms of a robustness evaluating Monte Carlo simulation consisting of 600 runs (see text for details). The x-axes show the error between the exact steady-state value of a message and its value as found by the spiking network. One hundred bins were used, and the histogram colors correspond to the colors of the messages in *b*. Curves corresponding to the messages sent by the nodes $P_i(Y_i \mid X_i)$ have been omitted because the underlying computations are simple and errors are not of relevant magnitude (comparable to the yellow graph in Figure 6a). (d) Correlation coefficients for each message in *c* between a numerical evaluation of continuous time BP and the population rates of the readout pools. External input was provided by instances of the stochastic process as defined in section 2.3. These time series had a length of 20 s in simulated time. The correlation coefficients were highly significant with p-values less than 1%.

within the BP framework, it is possible to obtain the marginal a posteriori probabilities $P(X_i \mid Y_1, \ldots, Y_4)$ for all i simultaneously, that is, the degree of belief one can have in the value of each bit at the beginning of the channel, given the observed values of those bits at the end of the channel. The messages are scalar values in [0, 1] representing the sender nodes' local estimate about the associated variable to take on value 1. An example message obtained by inserting the definition of the ⊕-factor into equation 2.1 can be found in the supplementary material.

A network of spiking neurons equivalent to the graph in Figure 3a is shown in Figure 3b. We evaluated the ability of this neuronal circuit to perform inference against a series ($n = 600$) of random external stimulations (corresponding to the messages along the variables $Y_1, \ldots, Y_4$). For each trial, all of these external inputs were assigned to a random but temporally constant value (constant instantaneous firing rates). Then for internally computed messages, the error between their exact steady-state value and the corresponding value found by the spiking network was determined (see Figure 3c). Each colored curve shows the histogram of the error $m_{correct} - m_{spiking}$ between the correct steady-state message($m_{correct}$) and the one found by the spiking network ($m_{spiking}$) during each run. The spiking messages are normalized to 1 such that they correspond to probabilities. The black histograms serve as control and correspond to the error $m_{correct} - m_{random}$; that is, in each run, $m_{spiking}$ has been replaced by a random probability m_{random}. A run lasted for 0.5 s in simulated time. For each message, steady state was assumed to be reached after 0.2 s, and the temporal averages of the instantaneous firing rates during the remaining 0.3 s were taken as the encoded steady-state solutions found by the spiking network. The numbers on top of each curve are the trial-averaged Kullback-Leibler (KL) divergences (in bits) between the distributions induced by $m_{correct}$ and $m_{spiking}/m_{random}$, respectively. The trial-averaged entropies (H) of the correct distributions were between 0.59 and 0.69 bits. If $m_{correct}$ and $m_{spiking}$ were both drawn from a uniform distribution of distributions, the expected KL divergence $\mathbb{E}(KL)$ and entropy $\mathbb{E}(H)$ would be given by $\mathbb{E}(KL) = \mathbb{E}(H) = \frac{1}{2 \cdot \ln(2)} \approx 0.72$ bits.

The results summarized in Figure 3c indicate a close correspondence between the BP steady-state solution and the solution found by the spiking network. However, the performance decreases with increasing depth of the messages: that is, the KL divergence and histogram width associated with messages that depend on strongly processed input are typically larger than those of messages that depend on less processed input (compare green, cyan/yellow, purple curves with pink/red curves, respectively). This effect could be caused by error propagation associated with the necessarily imperfect training of the readouts.

To gain further insight into the dependency between the dynamics as defined by equation 2.1 and those of the neural circuit, we also stimulated the circuit with spike trains of temporally varying instantaneous firing rates. These rates followed but were different instances of the same stochastic processes as those used during the training procedure (see section 2.3). Since the message dynamics can be important for the BP algorithm to converge (see section 2.1), the goal here was to assess the goodness of fit between a numerical solution of equation 2.1 and the spike dynamics of the readout units when random time series have been externally applied. The correlation coefficients between these two quantities for each message are summarized in Figure 3d. One can infer a strong linear relationship.

However as expected from the previous results, there is a slightly higher correlation for less processed input than for messages deep in the network. Also, coefficients associated with messages emitted by the $\oplus$-node are much smaller than those of messages emitted by the $=$-node. Probably this result arises because the sum-product rule of the former node is much harder to learn.

4.2 Application of the Model to a Psychophysical Inference Problem. Many probabilistic inference problems faced by biological systems can be described in an elegant way by using graphical models. For example, Kersten and Yuille (2003) and Kersten et al. (2004) define four basic types of Bayesian computations in the framework of object recognition and perception. One of these models the phenomenon of explaining away alternative hypotheses about the interpretation of a visual scene in the presence of auxiliary stimuli favoring a particular hypothesis. An example is shown in Figure 4a. Knill and Kersten (1991) have shown that when two cylinders with an identical photometrical luminance profile are attached to each other, subjects consistently perceive two identical objects. However, when the cylinders are replaced by cubes with the same luminance profile, the right cube as a whole appears to be brighter than the left one.

The authors explain these distinct perceptions as being due to the integration of auxiliary information provided by the contour shape of the two types of objects. The round contours of the cylinders make it more likely that the scene has a light source on the left, with constant material reflectance across the position (i.e., across the two cylinders). Thus, the two cylinders are interpreted and perceived as identical objects. This hypothesis explains away an alternative one that assumes uniform illumination and a higher reflectance of the right object. In the case of straight contours, the latter explanation is more likely, and hence the two cubes are perceived differently.

Figures 4b and 4c show a Bayesian network and its equivalent FFG, which provide a probabilistic interpretation of this phenomenon. Figure 4d shows our corresponding spiking neural network. The two nonprior factors of the FFG in Figure 4c ($P_1(S \mid R, O)$ and $P_2(C \mid O)$) are defined by the values given in Tables 1 and 2, respectively. As in the noisy channel case above, our goal was to implement general filters as expressed by equation 2.1 for arbitrary input signals $m_{k\to i}(t)$. An example message obtained by inserting the definition of the $P_1(S \mid R, O)$ factor into equation 2.1 can be found in the supplementary material.

We performed two types of simulations. First is a psychophysically relevant case, in which the visual scene in Figure 4a changes suddenly, so that curved contours are observed instead of straight ones (i.e., variable C in Figure 4 steps from 0 to 1). We expect this step to result in a decrease in the marginal probability of variable R (perceived reflectance), indicating the reduced belief in the homogeneous reflectance hypothesis.

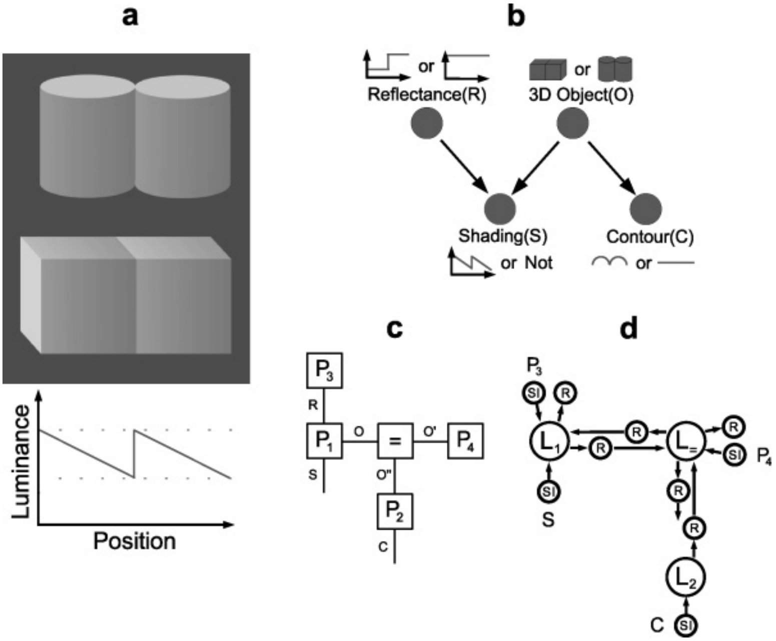


Figure 4: Graphical models and neural implementation of an explaining-away phenomenon. (a) Knill and Kersten's lightness illusion (see text for details; figure taken from Adelson & Gazzaniga, 2000; see http://web.mit.edu/persci/gaz/gaz-teaching/index.html for an animated movie making the illusion even more apparent). (b) Bayesian network describing the illusion in a as an explaining-away phenomenon. All variables (nodes) are binary. The observed luminance profile or shading can be either as in a ($S = 1$) or different ($S = 0$). The material reflectance of the two objects can be either homogeneous ($R = 0$) or such that the reflectance of the right object is higher ($R = 1$). There are two types of 3D objects, cylinders ($O = 1$) and cubes ($O = 0$), and two types of observed contours, curved ($C = 1$) and straight ($C = 0$). The network corresponds to the joint probability distribution $P(S, R, O, C) = P_1(S \mid R, O)P_2(C \mid O)P_3(R)P_4(O)$. In this model, whether two attached objects are perceived identically or distinctly will depend on the marginal probability of the material reflectance $P(R \mid S, C)$ given external observations S and C (graph taken slightly modified from Kersten et al., 2004). (c) FFG representing the same joint probability as in b. Observed variables S and C occur as singly connected edges. (d) Implementation of the BP algorithm with spiking neurons on the FFG shown in c. The individual liquid pools (L_i) correspond to the factors labeled by the same subscript i. Together with the associated spiking readouts (R) message, signals are implemented. Externally provided spike inputs (SI) stand for either observed variables (C and S) or constant messages originating from factors representing prior knowledge ($P_3(R)$ and $P_4(O')$). The purpose of the network is to compute all the (conditional) marginal probabilities associated with the unobserved variables.

Table 1: Factor Entries for $P_1(S \mid R, O)$.

a. $\boldsymbol{S = 0}$

	$\boldsymbol{O = 0}$	$\boldsymbol{O = 1}$
$\boldsymbol{R = 0}$	0.8	0.2
$\boldsymbol{R = 1}$	0.2	0.9

b. $\boldsymbol{S = 1}$

	$\boldsymbol{O = 0}$	$\boldsymbol{O = 1}$
$\boldsymbol{R = 0}$	0.2	0.8
$\boldsymbol{R = 1}$	0.8	0.1

Table 2: Factor Entries for $P_2(C \mid O)$.

	$\boldsymbol{O = 0}$	$\boldsymbol{O = 1}$
$\boldsymbol{C = 0}$	0.8	0.2
$\boldsymbol{C = 1}$	0.2	0.8

Figure 5 shows the results obtained by our spiking network and by direct numerical emulation of BP in that situation. There is a clear decline of the message along R (green trace) from about 0.6 to about 0.4. Since the prior of R was assumed to be uniform, this message corresponds directly to the (conditional) marginal probability of R. Therefore, after the transition, a constant reflectance across position ($R = 0$) is a more likely hypothesis than a reflectance step ($R = 1$). On a perceptual level, this result corresponds to a change in the interpretation of the scene; that is, the perception of two different objects switches to the perception of two identical objects. Note that the network computes the new steady-state probabilities in approximately 100 ms (interval between the gray vertical bars; the LSMs were trained with $\tau = 10$ ms in equation 2.1). This latency corresponds well to estimates of human performance in object classification tasks (which can also be seen as Bayesian inference problems) as reported in the psychophysics literature (Kirchner & Thorpe, 2006; Liu, Harris, & Kanwisher, 2002). With a classic single-neuron firing rate code such a fast performance would have been rather unlikely to occur, because in this case each computational stage (node) needs to perform temporal averaging of spikes, which slows down the dynamics more and more the deeper the node is in the graph.

As in the noisy channel case in the previous section, we also evaluated the ability of the neuronal circuit to perform inference against a series ($n = 600$) of random external stimulations. The results are summarized in Figure 6a. Once again the results indicate a close correspondence between the two solutions that is far from chance level. As before, we also evaluated

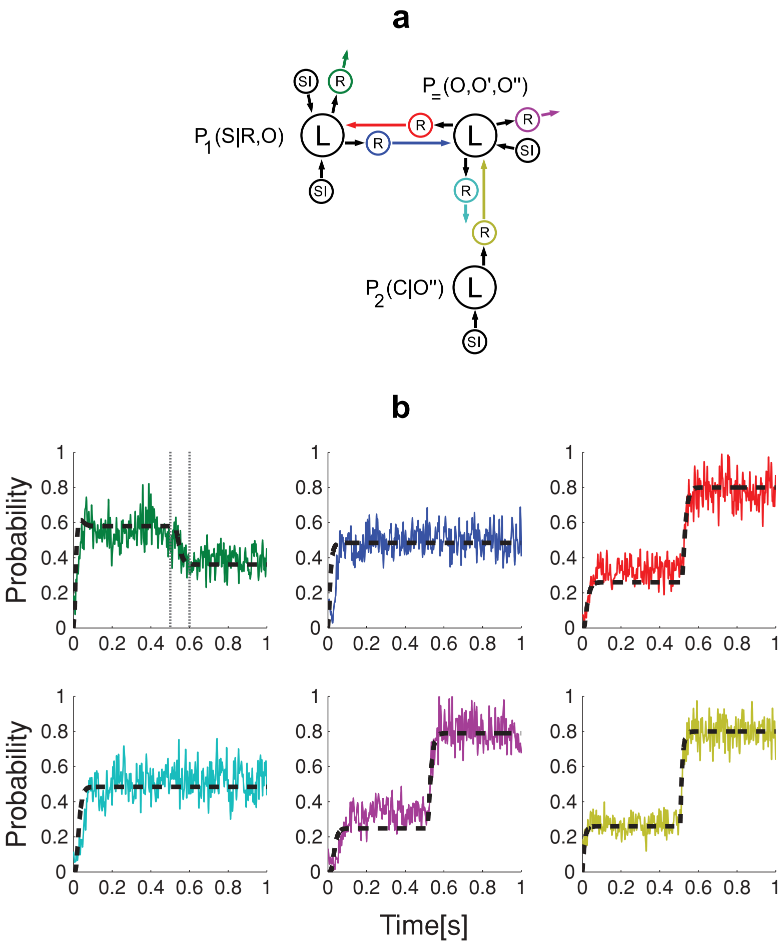
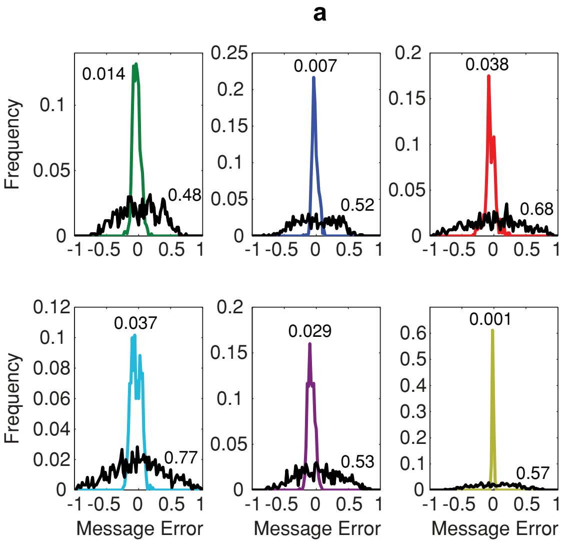


Figure 5: Simulating explaining-away with a sudden change in contour perception. (a) Color code of the messages (arrows) within the network of Figure 4. Externally supplied input messages are fed into the network via the black SI populations. All inputs were constant during the entire run besides the message entering node $P_2(C \mid O'')$ along observed variable C. This probability switched from 0.1 to 0.9 at time 0.5 s, indicating the visual scene has suddenly changed from observed straight to curved contours. External input messages from nodes representing prior knowledge were uniform ($m_{P_3(R)\to P_1(S|R,O)}(t) = m_{P_4(O')\to P_=}(t) = 0.5$), and a nonuniform luminance profile was observed ($m_{S\to P_1(S|R,O)}(t) = 0.9$) (b) Colored solid lines: Time series of the probability values encoded by the population rate of the individual readouts in response to the externally applied spike trains representing the input messages. Color code as in a. Black dashed lines represent the result of a nonneural simulation of BP by solving equation 2.1 numerically.



b

Message	$m_{P1\to P3}$	$m_{P1\to =}$	$m_{=\to P1}$	$m_{=\to P2}$	$m_{=\to P4}$	$m_{P2\to =}$
CC	0.73	0.85	0.94	0.97	0.94	0.99

Figure 6: (a) Performance analysis of the explaining-away circuit. Shown are the results of a Monte Carlo simulation consisting of 600 runs. In each run, all external spike inputs (SI in Figure 5a) were assigned to randomly chosen, constant values (constant instantaneous firing rates) corresponding to random input messages. Each colored curve shows the histogram of the error $m_{correct} - m_{spiking}$ between the correct steady-state message ($m_{correct}$) and the one found by the spiking network ($m_{spiking}$) during each run. Since the messages are normalized to 1, they correspond to probabilities. One hundred bins were used; the color code of the messages is as in Figure 5. Each run lasted for 0.5 s in simulated biological time. Steady state was assumed to be reached after 0.2 s, and the temporal averages of the instantaneous firing rates during the remaining 0.3 s were taken as the encoded steady-state solutions found by the spiking network. The black histograms serve as control and correspond to the error $m_{correct} - m_{random}$; that is, in each run, $m_{spiking}$ has been replaced by a random probability m_{random}. The numbers attached to each curve are the trial-averaged Kullback-Leibler divergences (in bits) between the distributions induced by $m_{correct}$ and $m_{spiking}/m_{random}$, respectively. The trial-averaged entropies of the correct distributions were between 0.48 and 0.67 bits. (b) Correlation coefficients for each message in *a* between a numerical evaluation of continuous time BP and the population rates of the readout pools. External input was provided by instances of the stochastic process as defined in section 2.3. These time series had a length of 20 s in simulated time. The correlation coefficients were highly significant with a p-value less than 1%.

the correlation coefficients between the spiking and the numerical solutions after applying random input time series, which gave the results of Figure 6b. There, we can also see a slight effect of error accumulation, which gives rise to smaller correlation coefficients for strongly processed messages (e.g., compare $m_{P1\to P3}$ to $m_{P1\to =}$). It can also be seen that for this graph, the minimum correlation coefficient of all messages is larger than in the noisy channel example. Most likely this is because the messages emitted by factor P_1 in the explaining-away graph were easier to learn than those emitted by the $\oplus$-node in the noisy channel case.

5 Discussion

As a proof of concept, we have shown that a network of spiking neurons can implement belief propagation on a factor graph. The principle is to use liquid state machines composed of pools of spiking neurons as function nodes of (Forney-style) factor graphs. Each pool gathers messages from its input nodes and combines these through its network dynamics. The various output messages transmitted over the edges of the graph are extracted from their source liquids by groups of readout neurons and fed into their destination liquid pools. We have applied this approach to an inference problem from the engineering domain dealing with the passing of bits through an unreliable signal transmission channel (see Figure 3) and also to a more biologically relevant case in which visual cues are used to resolve hypotheses about the interpretation of object shape and illumination (see Figure 4). The dynamics of these networks followed closely the numerical evaluation of the algorithm.

A restriction of our model is the use of binary variables. In this case, if normalized messages are assumed, each message (i.e., probability estimate) can be defined by a single scalar value, and so only a single readout population is needed to encode this value. In principle, though, our spiking readout method is extendable to vector-valued variables by using a population for each individual value of the underlying variable. Unfortunately, we expect this strategy to be expensive in terms of the number of required neurons for both the liquid and readout populations. A problem of a rate-based population code is that only the fraction of neurons firing during a small time interval, and not their position within the population, is the relevant variable. This restriction hinders the learning of the readouts, and we assume that it is part of the reason for the larger sizes of our liquid pools compared to those reported by Maass et al. (2002; see supplementary material). Therefore, we are currently applying a place-coding scheme to our approach, which is also closer to known properties (tuning curves) of real neurons.

We have chosen to implement the factor graph on a distributed arrangement of LSMs (see Figure 1) rather then on a single (global) liquid pool. However, due to the approximation properties of LSMs (Maass et al., 2002), one could construct a circuit consisting of just a single liquid pool, with each of its readouts feeding back into the pool in a recurrent manner. These readouts could also be trained according to the sum-product rule (see equation 2.1). Although this strategy looks plausible, there is a clear reason that the distributed arrangement is superior: in BP, every node uses exclusively locally available incoming messages to compute its outgoing messages. In the LSM framework, this means that each readout is required to access only a subset of all shared knowledge within the network. Using a single global liquid pool with recurrent readouts makes the state of the liquid neurons dependent on all information present in the network. Therefore, as the number of nodes in the graphical model increases, it becomes more difficult to filter out irrelevant information during training of the readouts.

The mechanism that we have presented here is very general and lends itself to implementation in, for example, the neuronal networks of the neocortex. Many anatomical studies indicate that the entire neocortex is composed of a relatively few basic types of excitatory and inhibitory neurons that are everywhere connected in a similar pattern, suggesting that cortex uses a common fundamental circuit architecture that repeats over its entire area (Mountcastle, 1997; Douglas & Martin, 2004). Various candidates for the computational process supported by this common architecture have been proposed (e.g. Hopfield, 1982; Pouget, Deneve, & Duhamel, 2002; Maass et al., 2002). But the ability to infer unknown information from incomplete sensory data combined with some prior knowledge must rank as one of the most fundamental principles for incorporation in cortical circuits. And in this letter, we have shown that such inference can be instantiated in a highly distributed manner, using a common architecture of interconnected pools of spiking neurons. In using pools of neurons, our approach stands in contrast to previous models whose basic inference elements are single neurons (Rao, 2004; Ott & Stoop, 2006; Deneve, 2007). The advantage of interacting pools is that they allow the implementation of FFGs, which belong to the most general types of graphical models. They subsume Bayesian networks (as in Rao, 2004; Deneve, 2007), as well as Markov random fields (as in Ott & Stoop, 2006). Bayesian networks and Markov random fields are restricted in that there are instances of graphs of either type that cannot be translated into the other while keeping the same set of variables (Bishop, 2006). For example, the noisy channel problem in Figure 3 has no analogous Bayesian network.

In our simulations, we have used general-purpose liquid pools, which, when interpreted biologically, opens the possibility for cortex to perform many other computations on top of and in parallel to BP using the same neural hardware. Therefore, our approach predicts that on the anatomical level, there is no specific neural circuitry implementing the sum-product rule. Instead, it can be just one out of many computational results that a specific readout can extract from a generic pool. For example, the most probable state of each marginalized variable connected to a given liquid pool might be determined by BP, and the resulting state vector can then be processed by a classifier readout unrelated to the inference task at hand. Such readouts could be represented by projection neurons transmitting their classification results to various target areas.

On the other hand, if cortex is a pure inference machine, a general-purpose liquid pool might be unnecessarily complex for only inference tasks. An interesting future direction of research is therefore the development of an appropriately structured liquid pool whose computational abilities are specialized to the supply of all necessary products according to the sum-product rule. Such a special-purpose circuit might be genetically prewired in cortex and can be thought of as an explicit multiplicative feature mapping whose scalar product with itself implements a kernel function as in the framework of support vector machines (Bishop, 2006). Therefore we consider a spiking version of the recently presented EVOLINO method (Schmidhuber, Wierstra, Gagliolo, & Gomez, 2007) as a suitable starting point for circuit creation by means of evolutionary algorithms. The sum-product rule would then be provided by the output of a trained readout through a linear combination of the circuit-provided products. In this context, a factor node is exclusively defined by the weights of its readouts, and a customization of the factor to a particular problem can then be accounted for by assigning specific values to those weights.

This approach raises the question of whether it is possible to learn these weights by unsupervised or semisupervised (reinforcement) learning methods. In this letter, we have not tackled this problem, as our intention was only to provide a proof of concept as proposed in Figure 1. As a next step, we will explore the use of reward-modulated STDP learning to obtain suitable readout weights (Legenstein et al., 2008; Izhikevich, 2007). This method has already been applied successfully to responses of a generic cortical microcircuit (liquid) model (Legenstein et al., 2008).

Acknowledgments

We cordially thank Matthias Bethge and Laurenz Wiskott for fruitful discussions about the proposed method and valuable suggestions regarding population coding. This work is funded by the EU within the DAISY and FACETS Grants (grant nos. FP6-2005-015803 and 15879).

References

Adelson, E. H., & Gazzaniga, M. (2000). *The new cognitive neurosciences* (2nd ed.). Cambridge, MA: MIT Press.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Berlin: Springer Science+Business Media, LLC.

Brunel, N., Chance, F. S., Fourcaud, N., & Abbott, L. F. (2001). Effects of synaptic noise and filtering on the frequency response of spiking neurons. *Physical Review Letters, 86*(10), 2186–2189.

Deneve, S. (2007). Bayesian spiking neurons 1: Inference. *Neural Computation, 20*(1), 91–117.

Deneve, S., Latham, P. E., & Pouget, A. (2001). Efficient computation and cue integration with noisy population codes. *Nature Neuroscience, 4*(8), 826–831.

Douglas, R. J., & Martin, K. A. C. (2004). Neuronal circuits of the neocortex. *Annual Review of Neuroscience, 27*, 419–451.

Ernst, M. O., & Banks, M. S. (2002). Humans integrate visual and haptic information in a statistically optimal fashion. *Nature, 415*, 429–433.

Gerstner, W., & Kistler, W. (2002). *Spiking neuron models*. Cambridge: Cambridge University Press.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA, 79*(8), 2554–2558.

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex, 17*, 2443–2452.

Kersten, D., Mamassian, P., & Yuille, A. (2004). Object perception as Bayesian inference. *Annu. Rev. Psychol., 55*, 271–304.

Kersten, D., & Yuille, A. (2003). Bayesian models of object perception. *Current Opinion in Neurobiology, 13*, 150–158.

Kirchner, H., & Thorpe, S. J. (2006). Ultra-rapid object detection with saccadic eye movements: Visual processing speed revisited. *Vision Research, 46*, 1762–1776.

Knill, D. C., & Kersten, D. (1991). Apparent surface curvature affects lightness perception. *Nature, 351*, 228–230.

Körding, K. P., & Wolpert, D. (2004). Bayesian integration in sensorimotor learning. *Nature, 427*, 244–247.

Kschischang, F. R., & Frey, B. J. (1998). Iterative decoding of compound codes by probability propagation in graphical models. *IEEE Journal on Selected Areas in Communications, 16*(1), 219–230.

Kschischang, F. R., Frey, B. J., & Löliger, H. A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory, 47*(2), 498–519.

Legenstein, R., Pecevski, D., & Maass, W. (2008). A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Computational Biology, 4*, e10000180.

Liu, J., Harris, A., & Kanwisher, N. (2002). Stages of processing in face perception: An MEG study. *Nature Neuroscience, 5*(9), 910–916.

Löliger, H. A. (2004). An introduction to factor graphs. *IEEE Signal Proc. Mag., 21*, 28–41.

Löliger, H. A., Dauwels, J., Hu, J., Korl, S., Ping, L., & Kschischang, F. R. (2007). The factor graph approach to model-based signal processing. *Proceedings of the IEEE, 95*(6), 1295–1322.

Ma, W. J., Beck, J. M., Latham, P. E., & Pouget, A. (2006). Bayesian inference with probabilistic population codes. *Nature Neuroscience, 9*(11), 1432–1438.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation, 14*, 2531–2560.

Mountcastle, V. B. (1997). The columnar organization of the neocortex. *Brain, 120*, 701–722.

Ott, T., & Stoop, R. (2006). The neurodynamics of belief-propagation on binary Markov random fields. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems, 18* (pp. 1057–1064). Cambridge, MA: MIT Press.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Francisco: Morgan Kaufmann.

Pouget, A., Deneve, S., & Duhamel, J. R. (2002). A computational perspective on the neural basis of multisensory spatial representations. *Nat. Rev. Neurosci., 3*(9), 741–747.

Rao, R. P. N. (2004). Bayesian computation in recurrent neural circuits. *Neural Computation, 16*(1), 1–38.

Schmidhuber, J., Wierstra, D., Gagliolo, M., & Gomez, F. (2007). Training recurrent networks by EVOLINO. *Neural Computation, 19*(3), 757–779.

Silberberg, G., Bethge, M., Markram, H., Pawelzik, K., & Tsodyks, M. (2004). Dynamics of population rate codes in ensembles of neocortical neurons. *Journal of Neurophysiology, 91*, 704–709.

von Helmholtz, H. (1867). *Handbuch der physiologischen Optik*. Leipzig: L. Voss.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2005). Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory, 51*(7), 2282–2312.

Yu, A. J., & Dayan, P. (2005). Inference, attention, and decision in a Bayesian neural architecture. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems, 17* (pp. 1577–1584). Cambridge, MA: MIT Press.

Received August 11, 2008; accepted January 17, 2009.