eNeuro

Research Article: Theory/New Concepts | *Cognition and Behavior*

Learning probabilistic inference through STDP

Storing uncertain information in microcircuit motifs

Dejan Pecevski and Wolfgang Maass

Institute for Theoretical Computer Science, Graz University of Technology, A-8010 Graz, Austria

DOI: 10.1523/ENEURO.0048-15.2016

Received: 5 April 2015

Revised: 25 January 2016

Accepted: 3 March 2016

Published: 25 March 2016

Conflict of Interest: The authors declare no competing financial interests; the submitted paper is in accordance with the Society's Policy on Conflict of Interest.

D.P. and W.M. designed research; D.P. analyzed data; D.P. and W.M. wrote the paper; D.P. and W.M. performed research

Correspondence should be addressed to: Wolfgang Maass, Graz University of Technology, Institute for Theoretical Computer Science, Inffeldgasse 16b/1, 8010 Graz, Austria; E-mail: maass@igi.tugraz.at

Cite as: eNeuro 2016; 10.1523/ENEURO.0048-15.2016

eNeuro

http://eneuro.msubmit.net

eN-TNC-0048-15R2

Learning probabilistic inference through STDP

1 Manuscript Title Page

1. Manuscript Title: Learning probabilistic inference through STDP

2. Abbreviated Title: Storing uncertain information in microcircuit motifs

3. List all Author Names and Affiliations in order as they would appear in the published article:
Dejan Pecevski, Institute for Theoretical Computer Science, Graz University of Technology, A-8010 Graz, Austria, E-mail: dejan@igi.tugraz.at
Wolfgang Maass, Institute for Theoretical Computer Science, Graz University of Technology, A-8010 Graz, Austria, E-mail: maass@igi.tugraz.at

4. Author Contributions:
Each author must be identified with at least one of the following: DP and WM Designed research; DP Performed research, DP and WM Wrote the paper.

5. Correspondence should be addressed to (include email address):
Graz University of Technology, Institute for Theoretical Computer Science, Inffeldgasse 16b/1, 8010 Graz, Austria; E-mail: maass@igi.tugraz.at

6. Number of Figures: 9

7. Number of Tables: 5

8. Number of Multimedia: none

9. Number of words for Abstract: 202

10. Number of words for Significance Statement: 116

11. Number of words for Introduction: 769

12. Number of words for Discussion: 2814

13. Acknowledgements:
We would like to thank Stefan Habenschuss for helpful comments on an earlier version of the manuscript.

14. Conflict of Interest:

    (a) No - The authors declare no competing financial interests; the submitted paper is in accordance with the Society's Policy on Conflict of Interest.

    (b) Yes (Please explain): n/a

15. Funding sources:
This paper was written under partial support by the European Union project FP7-269921 (BrainScaleS).

Learning probabilistic inference through STDP

Anonymous Authors

March 3, 2016

Abstract

Numerous experimental data show that the brain is able to extract information from complex, uncertain, and often ambiguous experiences. Furthermore it can use such learnt information for decision making through probabilistic inference. Several models have been proposed that aim at explaining how probabilistic inference could be carried out by networks of neurons in the brain. We propose here a model that can also explain how such neural network could acquire the necessary information for that from examples. We show that spike-timing-dependent plasticity (STDP) in combination with intrinsic plasticity generates in ensembles of pyramidal cells with lateral inhibition a fundamental building block for that: probabilistic associations between neurons that represent through their firing current values of random variables. Furthermore, by combining such adaptive network motifs in a recursive manner the resulting network is enabled to extract statistical information from complex input streams, and to build an internal model for the distribution p^* that generates the examples it receives. This holds even if p^* contains higher order moments. The analysis of this learning process is supported by a rigorous theoretical foundation. Furthermore we show, that the network can use the learnt internal model immediately for prediction, decision making, and other types of probabilistic inference.

Significance statement

Most memory models focus on pattern completion as computational operation for memory recall. But obviously our brains can also answer questions that involve diverse experiences in unexpected ways. Consider for example the question whether we went with Mr. X more often to lunch than to dinner, or whether Mr. X would probably like the new restaurant Z. These queries could not be anticipated during the formation of memory traces. But still we can answer such queries (which are special cases of probabilistic inference: estimation of a posterior marginal) without any effort. We propose a model based on experimentally observed rules for synaptic plasticity (STDP) which explains how a network of spiking neurons can acquire this capability.

Introduction

A large number of experimental data from neuroscience and cognitive science suggest that the brain carries out probabilistic inference for a large number of probability distributions p^* that describe different aspects of the environment of the organism and its interaction with the environment (Griffiths and Tenenbaum, 2006; Bonawitz et al., 2014). In models for probabilistic inference in networks of neurons one has so far (with the exception of (Siegelmann and Holzman, 2010), see Discussion) programmed the underlying distribution p^* into the architecture and parameters of the network model. However brains have to learn internal models p of salient distributions p^* in their environment from examples that are generated by p^*. We present in this article the first model that explains how networks of spiking neurons could learn such internal models p through experimentally supported local plasticity rules. Furthermore we show how a

network of spiking neuron can store the learnt information in a way that makes it readily accessible for probabilistic inference.

Two different approaches how networks of neurons in the brain could execute probabilistic inference have been proposed. One approach emulates an arithmetical (deterministic) algorithmic method for carrying out probabilistic inference through a suitable distributed organization of sums and products of probabilities (referred to as belief propagation or message passing), see e.g. (Steimer et al., 2009). The other approach for emulating probabilistic inference in networks of spiking neurons is based on the assumption that a network of neurons can "embody" a distribution p in such a way that it can generate samples from p. Probabilistic inference for p can then be carried out through simple operations on these samples, for example the computation of a posterior marginal just requires to look at the distribution of the random variable of interest in these samples. This approach is known in computer science as Markov-Chain-Monte-Carlo (MCMC) sampling, and is widely used to carry out probabilistic inference also for complex distributions p for which belief propagation approaches have no guarantee to provide correct answers. Whereas the previous approach prefers a deterministic network, where every stochasticity is detrimental for the performance, this second approach requires an inherently stochastic network of spiking neurons (Buesing et al., 2011). It has been argued that the dynamics of networks of neurons in the brain is in fact highly stochastic, both on the basis of inherently stochastic features of its components (such as stochastic synaptic release), and on the basis of trial-to-trial variability in neural recordings and observed variability in behavioral outputs.

The learning model that is presented in this article ties in to this second approach, and shows that stochastic networks of neurons are able to automatically absorb the relevant statistical information from examples that they receive. As a result, we have now one first complete theory for the emergence of probabilistic inference in networks of spiking neurons through learning. We focus in this article on somewhat idealized models for spiking neurons and plasticity rules, that allow us to give rigorous proofs for the convergence of learning.

We first show how an extension of an ubiquitous network motif of cortical microcircuits, interconnected populations of pyramidal cells with lateral inhibition (Winner-Take-All or WTA circuits, see (Douglas and Martin, 2004; Nessler et al., 2013)), gives rise to the basic building block for absorbing probabilistic information from examples. The output neurons of an array of WTA-circuits form the hidden layer of a 3-layer feedforward learning module, to which we refer as *stochastic association module*. We show that such a module can learn through STDP and plasticity of the excitability of neurons (intrinsic plasticity) stochastic associations between the random variables that are encoded by the firing of neurons on its first and third layer. The module can learn this probabilistic information from the statistics of activation and co-activation of these neurons when the network processes examples that are provided by its environment. An important second finding is that recursive combinations of this network module can learn even complex probabilistic relationships between large numbers of random variables. This network learning capability is in fact universal in the sense that the underlying theory implies a proof of principle that an approximation to any distribution p^* over discrete random variables can be learnt by exposing the network to examples drawn from p^*. In fact, one can show that if the network is too small or has too few connections for learning a close approximation of p^*, it will still strive towards approximating p^* as well as it can, given its limited resources. The understanding of this learning process is supported by the theory of Expectation Maximization (EM).

Results

Previous models for probabilistic inference in networks of spiking neurons have shown that one can program the parameters (e.g., conditioned probability tables) of a given distribution p^* over discrete random variables into a network of idealized models for spiking neurons, provided that the network has a suitable architecture. We provide in this article a proof of principle that these parameters of p^* do not have to be programmed into the network: they can be learnt by a network $\mathcal{N}$ of spiking neurons via simple local plasticity rules from examples $\tilde{\mathbf{y}}$ that are generated by p^*. This does not hold for every

neural network $\mathcal{N}$, but – like in any existing model for probabilistic inference in neural networks – only under suitable assumptions about the architecture of $\mathcal{N}$. This result provides a proof of principle that networks of neurons in the brain can not only carry out probabilistic inference for distributions p^* whose parameters are specified in the genetic code, but also for distributions p^* that an organism encounters in its environment.

The underlying theory of Expectation Maximization (EM) does not guarantee that p^* can be learnt perfectly. However it implies that a network $\mathcal{N}$ with a suitable architecture is expected to make progress in creating an increasingly more accurate internal model p for p^* when it receives more and more examples that are generated by p^*. EM does not guarantee that the internal model p converges to p^*, but it implies that the network learning process cannot "run around in circles" where p moves forth and back between better and worse approximations of p^*. This learning result is general insofar as it shows that internal models p can be learnt by a network $\mathcal{N}$ with a suitable architecture for external distributions p^* over any number of discrete random variables, with arbitrary – also higher order – dependencies among these random variables. But although the architecture of $\mathcal{N}$ will obviously have to depend on the number of random variables of p^*, we show that it suffices to assume that it consists of recursive interconnections of different copies of a simple generic network motif, to which we refer as a *stochastic association module*. This network motif is a 3-layer feedforward network of excitatory spiking neurons with lateral inhibition on the hidden layer (Fig. 2). We show that this simple microcircuit motif can be viewed as an atomic learning module, that extracts via STDP and intrinsic plasticity from examples probabilistic associations between input variables $\mathbf{x}$ and output variable z that are encoded through population coding on its input and output layer. An example is presented in Fig. 3. We will then show in the following subsection that this atomic learning module can be recursively combined to form a network that automatically approximates through STDP and intrinsic plasticity arbitrarily complex distributions p^* over many discrete random variables from examples generated by p^*. In other words, this network learns an internal probabilistic model p for its stochastic environment p^*. Furthermore this network has the property that it can readily apply this internal model by carrying out probabilistic inference for p through its inherent stochastic dynamics. Examples are presented in Fig. 6 – 9.

The neurons in our models are stochastic integrate-and-fire neurons which have been shown to match biological data quite well (Jolivet et al., 2006; Mensi et al., 2011; Gerstner et al., 2014). We assume that a neuron has at any time t the instantaneous firing probability density $\rho(t) = \frac{1}{\tau}\exp(u(t))$, where $u(t)$ is its membrane potential and τ is a time constant. When it fires a spike, the neuron enters an absolute refractory period of duration τ after which it resumes its stochastic firing. The membrane potential $u(t) = \sum_i w_i\epsilon_i(t) + b$ is assumed to be the sum of the PSPs $\epsilon_i(t)$ elicited by the spikes from its presynaptic neurons, where w_i is the synaptic efficacy of the i-th synapse (and b is the bias of the neuron). The theoretically best tractable shape of a PSP $\epsilon_i(t)$ would be a step function of length τ. But we show in examples 1 and 2 that the relevant learning properties also hold for $\boldsymbol{\alpha}$-shaped EPSPs that are commonly considered in theoretical neuroscience. On the side, we would like to point out that for biological neurons the EPSPs vary from shapes with a pronounced initial peak to shapes with smooth hills in dependence of the distance of the synapse to the soma (Williams and Stuart, 2002), and obtain yet other shapes if amplified through NMDA- or Ca- spikes (Larkum et al., 2009).

We employ a simple STDP rule, which has the advantage of being theoretically tractable. Let w be the weight of the synapse at the connection from some presynaptic neuron ν_{pre} to a postsynaptic neuron ν_{post}. At each postsynaptic spike of neuron ν^{post} at time t this weight undergoes an update $w \leftarrow w+\eta\Delta w$, where η is the learning rate and

$$\Delta w = \begin{cases} e^{-(w+w_-)} - 1, & \text{if } \nu_{pre} \text{ fired in } [t-\tau, t] \ , \\ -1, & \text{if } \nu_{pre} \text{ did not fire in } [t-\tau, t]. \end{cases} \tag{1}$$

The parameter w_- is a baseline parameter in the learning rule, and τ is a parameter that corresponds to the duration of postsynaptic potentials (PSPs). Fig. 1A shows the resulting STDP curve. The rule exhibits LTP only for pre-before-post spiking within a time window of duration τ, otherwise it exhibits LTD. The causal part of the STDP window curve has the same shape as the PSP kernel, which is similar to other theoretically derived plasticity rules (Toyoizumi et al., 2005; Pfister et al., 2006). The properties

of this plasticity rule were studied in (Nessler et al., 2013). It was shown there that it supports learning of an internal probabilistic model of the inputs in a WTA network. It was also shown there that the weight dependence of Δw in (1) fits quite well to experimental data. Fig. 1B shows that the shape of the STDP curve according to (1) looks like the commonly considered one when applying an intermediate pairing frequency of 20 Hz (see (Sjöström et al., 2001) for experimental data on the dependence of the shape of the STDP curve on the pairing frequency).

It is well known that the excitability of neurons changes in dependence of their history of firing activity (Daoudal and Debanne, 2003; Cudmore and Turrigiano, 2004). We model this intrinsic plasticity of neurons through a simple rule according to which the bias b of a neuron changes at each spike of the neuron according to $b \leftarrow b + \eta' \, \Delta b$, with

$$\Delta b = \tau e^{-(b+b_-)} \quad , \tag{2}$$

where η' is a learning rate and b_- is a baseline parameter. In addition, we assume that the bias exhibits constant decay according to the differential equation

$$\frac{db}{dt} = -\eta' \quad . \tag{3}$$

A network module for learning stochastic associations

The atomic learning module in our model is a simple microcircuit motif that learns associations between some array of random variables $\mathbf{x} = (x^1, ..., x^I)$ and another random variable z from examples $\langle \mathbf{x}, z \rangle$ that are presented to the network. The variables $\mathbf{x}$ and z could for example represent different higher level features of an image. Or the variables $\mathbf{x}$ could represent higher level features of some visual stimulus, and the variable z a feature of a simultaneously occurring auditory stimulus. More formally, we assume that the network is exposed to examples $\langle \mathbf{x}, z \rangle$ consisting of concrete assignments of discrete values to the variables $\mathbf{x}$ and z, that are drawn from some unknown distribution $p^*(\mathbf{x}, z)$. We want to determine under what conditions a network module is able to create autonomously from exposure to these examples an internal model $p(\mathbf{x}, z)$ for $p^*(\mathbf{x}, z)$, that approximates p^* when the number of examples grows. Note that in general the *same* input $\mathbf{x}$ will occur in combination with *different* values $z(1), z(2), \ldots$ of z in the training examples, and the goal of learning is to learn for each value $z(i)$ the *probability* that it occurs for input $\mathbf{x}$. Hence the learning performance will not be evaluated by counting errors, i.e., deviations from a target output value. Rather, it will be evaluated by how well the network reproduces for any input value $\mathbf{x}$ the *distribution* of output values z.

We show that a 3-layer network of spiking neurons with the architecture shown in Fig. 2 can accomplish this learning task through STDP on synaptic connections from the first to the second layer and intrinsic plasticity of excitatory "hidden" neurons $\boldsymbol{\alpha}$ on the second layer. The weights of synaptic connections between the second and third layer are assumed to be fixed. These weights are assumed to have a large value, so that the firing of a neuron α on the second layer causes with very high probability the firing of the neuron on layer 3 to which it is connected.

We assume that each of the random variables $\mathbf{x}$ and z is represented by a population of neurons ("population coding"), with each value of the variable encoded by a separate neuron in the population, as indicated at the top of Fig. 2 for the variables x^i, and at the bottom for the variable z. The firing of a particular neuron χ^{im} in the population coding of variable x^i encodes the fact that x^i assumes the value m in the currently presented example. Similarly the firing of neuron ζ^l in the population $\boldsymbol{\zeta}$ for the variable z encodes the value l of this variable (see Fig. 2). Finally, we assume that an example $\langle \mathbf{x}, z \rangle$ is presented to this 3-layer network during learning in the following manner: Those neurons in the first layer that represent the given values of the variables $\mathbf{x}$ are made to fire at a high rate, whereas the other neurons in the first layer are inhibited and kept silent. In addition, if variable z has value l in this example, all hidden neurons outside the corresponding subpopulation $\boldsymbol{\alpha}^l$ are inhibited, so that they cannot fire (actually, it would suffice to block STDP for these neurons). An alternative view is that only a selected subset of neurons is disinhibited. New experimental data (see (Caroni, 2015) for a review) suggest that

in fact inhibition of synaptic plasticity (especially via Somatostatin-positive neurons) and disinhibition via VIP interneurons (Letzkus et al., 2015) play an important role in the control of plasticity in cortical microcircuits.

When the network does not receive examples, the neurons in this network module fire according to the stochastic dynamics of the model, and no plasticity is assumed to occur. We present in Methods a rigorous proof that after learning the distribution of output values z for a given network input $\mathbf{x}$ approximates in this stochastic association module the conditional distribution $p^*(z|\mathbf{x})$ of the joint distribution $p^*(\mathbf{x}, z)$ from which the values of z and $\mathbf{x}$ are drawn in the training examples. In fact, one can view this module from a theoretical perspective as an implicit generative model $p(\mathbf{x}, z; \boldsymbol{\theta})$ for the examples $\langle \mathbf{x}, z \rangle$, and one can prove that the network module carries out a stochastic search (stochastic online Expectation Maximization) that strives to minimize the Kullback-Leibler divergence $D_{KL}(p^*(\mathbf{x}, z)||p(\mathbf{x}, z; \boldsymbol{\theta}))$ between the external distribution p^* from which the examples are drawn and its internal model $p(\mathbf{x}, z; \boldsymbol{\theta})$ (see Theorem 1 and 1^* in Methods). The implicit generative model of the module is encoded in the synaptic weights and biases of the $\boldsymbol{\alpha}$ neurons.

Since the learning module represents the full joint distribution $p(\mathbf{x}, z; \boldsymbol{\theta})$, not just the conditional distribution $p(z|\mathbf{x}; \boldsymbol{\theta})$, it is the joint distribution that is considered to be the internal model of the learning module. This is more than just representing $p(z|\mathbf{x}; \boldsymbol{\theta})$ as the module also represents the distribution $p(\mathbf{x}; \boldsymbol{\theta})$. The distribution $p(\mathbf{x}; \boldsymbol{\theta})$ is represented in a sense that all probability values $p(\mathbf{x}; \boldsymbol{\theta})$ for each value of $\mathbf{x}$ can be calculated from the synaptic weights and biases of the $\boldsymbol{\alpha}$ neurons. This can be done by first calculating $p(\mathbf{x}, z; \boldsymbol{\theta})$ for each value of z based on the probabilistic model (see Methods), and then marginalizing out z . Another reason why $p(\mathbf{x}, z; \boldsymbol{\theta})$ is considered as internal model is that the learning rules are based on minimizing the Kullback-Leibler divergence between the internally represented joint distribution $p(\mathbf{x}, z; \boldsymbol{\theta})$ and the target distribution $p^*(\mathbf{x}, z)$ of the examples. In other words, the module implements a generative model learning. The conditional distribution becomes important after learning, when the module performs its function realized through the firing of the output neurons that approximates $p^*(z|\mathbf{x})$. This functional property of the learning module enables composing networks of modules that can learn larger distributions, as described in section "Recursive combinations of the basic learning module enable efficient learning of complex distributions from examples" of Results.

One may wonder why a 2-layer network would not suffice for learning such stochastic associations between random variables $\mathbf{x}$ and z. The simplest approach would be a model without hidden neurons, where the strengths of the synaptic connections between the neurons in the population codes for $\mathbf{x}$ and z encode the probability that a vector $\mathbf{x}$ is encountered in conjunction with a particular value of z. But this approach would restrict very much the types of internal models $p(\mathbf{x}, z)$ that the network could learn. In particular, it could not handle a situation where the distribution $p^*(\mathbf{x}, z = l)$ is multi-modal, i.e. when there are multiple modes in the distribution of $\mathbf{x}$ that are likely to occur in conjunction with a specific value l of z. For example in Fig 3B for $z = 2$, the distribution $p^*(\mathbf{x}, z = 2)$ has two modes, i.e. $x^1 = 1$ can occur in combination with $x^2 = 2$, and $x^1 = 2$ in combination with $x^2 = 1$ (whereas the assignments where $x^1 = x^2$ do not occur). The reason for this restriction to unimodal distributions is that the neuron ζ^l that represents $z = l$ in the population code for z would have to represent through the implicit generative model that is defined by the weights of afferent synapses and its excitability the marginal distribution $p^*(\mathbf{x}, z = l)$. But a single linear neuron can only represent one mode of a probability distribution of $\mathbf{x}$. However if one considers more complex neuron models with nonlinear dendritic processing, they can in principle also represent multi-modal distributions (see for example Fig. 4 and 5 in (Pecevski et al., 2011) and (Legenstein and Maass, 2011)). Hence with such more complex neuron models a more shallow learning network could potentially be used as a learning module in our architecture.

The 3-layer circuit in Fig. 2 can be viewed as a minimal model for allowing multi-modal distributions of $\mathbf{x}$ to be associated with a value of z. In fact, if one allows sufficiently many hidden neurons $\boldsymbol{\alpha}$, this representation becomes arbitrarily precise. These hidden neurons $\boldsymbol{\alpha}$ represent *combinations* of features represented through the neurons $\boldsymbol{\chi}^i$ that encode the variables $\mathbf{x}$. This mixed coding is reminiscent of experimental data on neurons in the cortex, see e.g. (Rigotti et al., 2013; Mante et al., 2013) .

We exploit here a generic property of STDP in WTA circuits, that was made explicit in (Nessler et al., 2013; Habenschuss et al., 2013b): If the neurons in the populations for the variables x^i are synaptically

connected to a set of neurons $\boldsymbol{\alpha}$ in a WTA circuit, and these synaptic connections are subject to STDP, then the neurons $\boldsymbol{\alpha}$ learn automatically a multi-modal internal model for the distribution of the variables $\mathbf{x}$. The learned probabilistic model is a mixture of multinomials. More precisely, each WTA neuron α specializes to fire in response to input patterns from one mode of $p^*(\mathbf{x})$. This specialization is produced by the plasticity rules (1), (2) which, when a neuron fires in response to some input pattern, adapt the weights and biases of the neuron so that in the future it fires with higher probability in response to the same pattern. At the same time, the competition enforced by the lateral inhibition between the $\boldsymbol{\alpha}$ neurons tends to prevent that multiple WTA neurons specialize on the same mode of $p^*(\mathbf{x})$.

This emergent property of STDP in WTA circuits was considered in (Nessler et al., 2013; Habenschuss et al., 2013b) in a setting where no association of $\mathbf{x}$ with other variables z needed to be learnt. In order to learn associations with z, we apply this mechanism in parallel for every possible value of z. In particular, we assume that in the population $\boldsymbol{\alpha}$ of WTA neurons there are disjoint subpopulations $\boldsymbol{\alpha}^l$ for each possible value l of z. The subpopulation $\boldsymbol{\alpha}^l$ projects to the ζ^l neuron with strong synaptic weights so that a spike of a neuron in $\boldsymbol{\alpha}^l$ causes also the neuron ζ^l to fire (see Fig. 2). As the WTA subcircuit $\boldsymbol{\alpha}^l$ is allowed to fire only for examples from $p^*(\mathbf{x}|z=l)$ it learns to approximate this distribution.

Intrinsic plasticity of the excitability of the hidden neurons $\boldsymbol{\alpha}^l$ is also essential for successful learning. As they are not firing during a presentation of example $\langle \mathbf{x}, z\rangle$ with $z \neq l$, the only adaptation in them for such example is a decay of their intrinsic excitabilities (see (3)). This supports learning of a representation of the marginal probability $p^*(z=l)$ in the biases of the neurons in $\boldsymbol{\alpha}^l$. Hence, the population $\boldsymbol{\alpha}^l$ learns a probabilistic model $p(\mathbf{x}, z=l;\boldsymbol{\theta})$ of the target probability distribution $p^*(\mathbf{x}, z=l) = p^*(\mathbf{x}|z=l)p^*(z=l)$. In this way all populations $\boldsymbol{\alpha}^l$ together learn a generative model $p(\mathbf{x}, z;\boldsymbol{\theta})$ for the joint distribution $p^*(\mathbf{x}, z)$ of the presented examples.

Further details can be found in the section "Theoretical properties of the basic learning module (stochastic association module) and its plasticity" of Methods.

Example 1: The learning module extracts complex stochastic associations from examples

We illustrate the inner workings of the learning module in an example where the task is to learn an internal model of an example target distribution $p^*(x^1, x^2, z)$ over binary RVs. This learning task is nontrivial since the distributions of values of $\mathbf{x}$ that are stochastically associated with the values $z=2$ and $z=1$ according to p^* are multi-modal (see gray bars in Fig. 3B). After learning, the output neurons of the module should fire for input $\mathbf{x}$ according to the conditional probability $p^*(z|\mathbf{x})$. The structure of the network module is depicted in Fig. 3A. It has two hidden neurons in the populations $\boldsymbol{\alpha}^1$ and $\boldsymbol{\alpha}^2$, which learn the two modes of $p^*(\mathbf{x}, z=1)$ and the two modes of $p^*(\mathbf{x}, z=2)$ respectively (see Fig. 3B). The learning period of the module lasted 1200s of simulated biological time. During learning, examples from $p^*(\mathbf{x}, z)$ were presented to the module for 100ms each. After learning, each WTA subcircuit $\boldsymbol{\alpha}^l$ had in fact acquired an approximation of the distribution $p^*(\mathbf{x}, z=l)$, as can be seen in Fig. 3B. The learning of the internal model in the WTA subcircuit $\boldsymbol{\alpha}^2$ is achieved through a process where each hidden neuron specializes to represent one of the two modes of $p^*(\mathbf{x}, z=2)$ that are shown in Fig. 3B. For the subcircuit $\boldsymbol{\alpha}^1$ the results are similar (not shown). The learning of an approximation to $p^*(\mathbf{x}, z)$ as an internal model automatically produces an approximation of the conditional $p^*(z|\mathbf{x})$ by the firing probabilities of the output neurons (see Fig. 3D). A typical resulting firing pattern is shown in Fig. 3F.

Recursive combinations of the basic learning module enable efficient learning of complex distributions from examples

The stochastic association module shown in Fig. 2, 3 is self-consistent in the sense that the input variables x^i are encoded through population coding in the same way as the output variable z. Hence one can recursively combine these modules so that the output population of one module becomes part of the input population of another module (see Fig. 4). The resulting more complex network is then not only able to learn a single probabilistic association between RVs, but many such associations simultaneously.

The basic learning modules form here not only chain connections, but typically also cycles, where the RV that is the output of the second module is simultaneously an input to the first module in the chain (like the variable y^k in Fig. 4).

Elementary results from probability theory imply that such recursive combinations of probabilistic associations between RVs have a very powerful, in fact universal, representation capability: the dependency structure of *every* probability distribution p^* over discrete RVs can be represented as a network of probabilistic associations between each of the RVs y^k and a subset of the other RVs (Bishop, 2006). More precisely, in the representation of an arbitrary distribution p^* one has a subnetwork (module) for each RV y^k of p^* that has y^k as output variable and the random variables in the Markov blanket $\mathbf{y}^{B(k)}$ of y^k as input variables. The Markov blanket defines a set of random variables so that conditioned on their values, y^k becomes independent from all remaining variables. For example, if p^* can be represented by a Bayesian network, it suffices to include in $\mathbf{y}^{B(k)}$ the parents of y^k together with the children of y^k, and their co-parents.

Whereas the classical results from probability theory only imply that one can represent any distribution p^* over discrete RVs as such recursive network of probabilistic associations, it was shown in (Buesing et al., 2011; Pecevski et al., 2011) that any such target distribution p^* can also be represented as stationary distribution of a network of spiking neurons, if suitable parameters (weights and biases) are programmed into the network. One only needs to assume that every spike of a neuron that participates in the population coding for one of the RVs y^k sets the value of y^k for a time period of length τ (= standard length of an EPSP) equal to the value encoded by this neuron. Then a suitably programmed network $\mathcal{N}$ of spiking neurons that results from recursive combinations of the basic module from Fig. 2 can represent any distribution p^* through its spontaneous firing activity (provided that each module for a RV y^k represents $p^*(y^k|\mathbf{y}^{B(k)})$ as described above). If one decodes the current firing activity in the network $\mathcal{N}$ at any time t by setting each RV y^k to that value that is indicated by the most recent firing of a neuron in the population code for y^k, the resulting distribution of value assignments to the RVs $y^1, \ldots, y^K$ of p^* over time is exactly the one given by p^*. In other words: p^* is the stationary distribution of the Markov chain that is defined by this network $\mathcal{N}$ of stochastically firing neurons. On the side, we would like to point out that this holds only after some initial "burn-in" phase, during which the distribution of network states becomes independent of the initial network state (see (Habenschuss et al., 2013a) for details).

We now show (see Fig. 5), that if one takes the previously analyzed learning capability of the basic network modules $\mathcal{N}^k$ into account, the composed spiking network $\mathcal{N}$ learns from examples $\tilde{\mathbf{y}}(n)$ of value assignments to $\langle y^1, \ldots, y^K \rangle$ drawn from p^* values $\boldsymbol{\theta}$ for its weights and biases that provide an approximation $p(\mathbf{y}; \boldsymbol{\theta})$ of $p^*(\mathbf{y})$. This approximation $p(\mathbf{y}; \boldsymbol{\theta})$ is represented by the network in the form of its stationary distribution of network states that result from its spontaneous firing activity. In order to achieve that, one just needs to allow each learning module $\mathcal{N}^k$ to learn in parallel from those components of the example $\tilde{\mathbf{y}}(n)$ that concern the random variables that it represents in the previously described manner. More precisely, each module $\mathcal{N}^k$ receives the components $\langle \tilde{y}^k(n), \tilde{\mathbf{y}}^{B(k)}(n) \rangle$ of each example $\tilde{\mathbf{y}}(n)$ that is presented to the network (for $n = 1, 2, \ldots$). The network $\mathcal{N}$ learns an approximation of p^* from examples $\tilde{\mathbf{y}}(n)$ without any additional computational overhead or teaching signals. Each subnetwork $\mathcal{N}^k$ learns through STDP and intrinsic plasticity an internal model $p^k(y^k, \mathbf{y}^{B(k)}; \boldsymbol{\theta}^k)$ of the marginal distribution $p^*(y^k, \mathbf{y}^{B(k)})$, as described in the preceding section.

One can rigorously prove that the sum over k of Kullback-Leibler divergences between the marginal distributions $p^*(y^k, \mathbf{y}^{B(k)})$ and the learnt internal models $p^k(y^k, \mathbf{y}^{B(k)}; \boldsymbol{\theta}^k)$ converges through these adaptive processes to a local minimum (see Theorem 2 in Methods). In this sense the spiking network $\mathcal{N}$ learns an approximation $p(\mathbf{y}; \boldsymbol{\theta})$ of the distribution $p^*(\mathbf{y})$.

As long as the RVs $\mathbf{y}$ are not split into inputs and outputs, this learning process is a typical example of unsupervised learning (see the definition in standard textbooks, such as (Bishop, 2006; Murphy, 2012; Haykin, 2009)). Characteristic for unsupervised learning is that learning progress is measured in terms of the deviation between the learnt distribution $p(\mathbf{y}; \boldsymbol{\theta})$ (= learnt internal model) and the distribution $p^*(\mathbf{y})$ from which the examples are generated. One also refers to this type of learning as density estimation.

Unsupervised learning has previously already been studied in a large number of artificial neural networks models: from Boltzmann machines (Ackley et al., 1985), neural belief networks (Neal, 1992), up

to deep learning networks (Salakhutdinov and Hinton, 2012; Bengio et al., 2015). One major motivation of this work has been to discover learning principles of the brain, based on the argument that supervision for learning is rare in the brain. Our learning model provides a complementary approach, with the main difference being that it is based on networks of spiking neurons, rather than artificial neural networks, and that it uses STDP as primary plasticity mechanism. A major difference between the learning process in Boltzmann machines and our model is that our model does not require separate sleep phases. Its learning process is more similar to parameter learning in Bayesian networks (see ch. 17 in (Koller and Friedman, 2009)). There the learning process also amounts to learning for each RV separately and in parallel from examples the conditional probability table for each RV, conditioned on the values of its parents. Such learning of a conditional probability table is analogous to the learning in a stochastic association module, except that such association module considers all RVs in the Markov blanket of a given RV, rather than just its parents. However our learning approach is more general than parameter learning in Bayesian networks insofar, as it also encompasses aspects of structure learning (ch. 18 in (Koller and Friedman, 2009)), see the discussion below in the section "Small numbers of hidden neurons in the learning modules often suffice".

Like other generative models for unsupervised learning, our model also aims at extracting underlying structure in the training examples (Hinton et al., 1995), so that it can even generate fake examples that share the discovered underlying structure (see Fig. 7). On the level of higher cortical areas such unsupervised learning could detect relationships between different types of features (see Fig. 6-9), between object representations in different sensory modalities, or how an action modifies the environment.

In contrast to the previously mentioned paradigms for unsupervised learning in neural networks, and similar to parameter learning in Bayesian networks, the architecture that we are proposing has a clear modular structure (see Fig. 5). It consists of stereotypical network motifs $\mathcal{N}^k$ that each try to determine for one of the RVs y^k to what extent values for y^k can be predicted from the values of other RVs (more precisely: the RVs in its Markov blanket $\mathbf{y}^{B(k)}$). As soon as this prediction becomes better than chance, the learning module $\mathcal{N}^k$ has discovered some underlying structure in the examples $\tilde{\mathbf{y}}$. One curious feature of this local prediction learning is that the learning process looks from the perspective of the learning module $\mathcal{N}^k$ like supervised learning, since $\tilde{y}^k$ is the prediction target for input $\tilde{\mathbf{y}}^{B(k)}$ to this module, and both $\tilde{y}^k$ and $\tilde{\mathbf{y}}^{B(k)}$ are part of a training example $\tilde{\mathbf{y}}$. This holds in spite of the fact that the whole examples $\tilde{\mathbf{y}}$ are in general presented to the network without any supervision, i.e. without any associated target output. This feature of the learning process is shared with parameter learning in Bayesian networks, where the learning of a conditional probability table for a RV y may look locally like supervised learning, because both the values of its parent nodes and the value of y are extracted from each training example.

Boltzmann machines and probabilistic graphical models such as Bayesian networks that are usually trained through unsupervised learning can however also be used for supervised learning (see e.g. (Hinton et al., 2006)). In that case the RVs $\mathbf{y}$ are split into two subsets $\mathbf{y}_I$ and $\mathbf{y}_O$, i.e. the target output $\tilde{\mathbf{y}}_O$ is combined with the vector $\tilde{\mathbf{y}}_I$ to form the examples $\tilde{\mathbf{y}} = \langle \mathbf{y}_I, \mathbf{y}_O \rangle$ used for training. The goal is to learn a mapping from $\mathbf{y}_I$ to $\mathbf{y}_O$, i.e. to learn the distribution $p^*(\mathbf{y}_O|\mathbf{y}_I)$. We consider here the general case where the mapping from inputs to target values is stochastic, either due to present noise in the target values or their inherent stochastic relation to the inputs. Typical classification problems where the mapping is assumed to be a deterministic function represent a special case of the stochastic formulation. For example for supervised learning of image categorization in Boltzmann machines one simply adds the target category like an additional feature to the feature vector of an image. The learning process remains exactly the same (i.e., unsupervised learning from examples $\tilde{\mathbf{y}}$), and for learning one does not have to tell the network which variables are inputs and which are outputs. The only difference is that after learning only the components $\mathbf{y}_I$ of new test examples are provided and the network has to produce a guess for values of the components $\mathbf{y}_O$. The same principle applies to the neural network in our approach: If $\mathbf{y}$ is partitioned into $\mathbf{y}_I$ and $\mathbf{y}_O$ it can learn the input-output mapping by learning an internal model of the full joint distribution $p^*(\mathbf{y}_I, \mathbf{y}_O)$ of the examples. After the learning process has finished, the network can estimate the probabilities of the target values $p^*(\mathbf{y}_O|\mathbf{y}_I)$ given a particular input $\mathbf{y}_I$. But note that, as it learns the joint distribution, the network can additionally also answer any other probabilistic inference queries based on the probability distribution p^* of the examples (see next section "Flexible retrieval of

learnt statistical information through probabilistic inference" of Results).

Classical learning of associative memory (like in a Hopfield network) also appears as a special case of the type of network learning that we are investigating: If $p^*(\mathbf{y})$ is nonzero only for some set $\tilde{\mathbf{y}}(0), \ldots, \tilde{\mathbf{y}}(M-1)$ of M memory items. For these memory items the network learns then input completion. Such input completion can be accomplished through the learnt internal model $p(\mathbf{y}; \boldsymbol{\theta})$ in our framework: If one clamps some of the neurons to the values of some given incomplete pattern $\hat{\mathbf{y}}_I$, and lets the other neurons fire according to the stochastic dynamics of this internal model. But note that the learning task of our model is more demanding than classical learning of associative memory, since it also has to learn the probability distribution (frequency) of the patterns $\tilde{\mathbf{y}}(n)$.

The underlying learning theory for our model (based on Expectation Maximization) does not guarantee that the internal model $p(\mathbf{y}; \boldsymbol{\theta})$ converges to the target distribution $p^*(\mathbf{y})$. Rather, as in all known cases of nontrivial unsupervised learning and self-organization it can only guarantee that a local optimum is achieved (which cannot get worse if learning continues). However, even this weak form of a theoretical guarantee is actually quite rare in the literature on neural network learning via STDP. Most known successful methods for unsupervised learning or self-organization in machine learning are supported theoretically in the same weak manner via Expectation Maximization. However many of these methods work very well in practice. The learning speed and quality depend for the learning framework that we have introduced on the nature of the target distribution p^*. We are demonstrating in Example 2 that this learning scheme works well for an example of p^* where it is well known that humans are able to learn a good approximation of probabilistic inference for p^*.

Flexible retrieval of learnt statistical information through probabilistic inference

After learning, the network $\mathcal{N}$ from Fig. 5 has an approximation $p(\mathbf{y}; \boldsymbol{\theta})$ to $p^*(\mathbf{y})$ as its stationary distribution. This holds under the convention that the firing of a neuron that represents a specific value l of a variable y^k sets this variable to value l for a duration τ equal to the duration of a generic EPSP. The learned distribution is manifested in the spontaneous activity of the network, i.e. when no neurons in the network are clamped. Information can be extracted from this learnt internal model $p(\mathbf{y}; \boldsymbol{\theta})$ through probabilistic inference via neural sampling. This type of information retrieval goes far beyond input completion, which is the only form of information retrieval in classical neural network models for memory. In particular, after inserting evidence into $\mathcal{N}$ by exciting or inhibiting some of the neurons that represent a subset $\mathbf{y}_e$ of the variables, the spiking activity of the rest of the network generates according to (Pecevski et al., 2011) samples from the conditional posterior distribution $p(\mathbf{y}_s|\mathbf{y}_e; \boldsymbol{\theta})$, where $\mathbf{y}_s$ is the subset of variables that are not in the $\mathbf{y}_e$. Furthermore, the posterior marginal probabilities $p^*(y^k|\mathbf{y}_e)$ of the variables y^k in $\mathbf{y}_s$ can be read out from the resulting firing rates of the neurons that represent these variables. Thus information gathered from the examples $\tilde{\mathbf{y}}(n)$ that had been presented to the network $\mathcal{N}$ (see Fig. 5) can be extracted from this network in very flexible ways through probabilistic inference. This will be demonstrated for an example in Fig. 6-9. In particular, the network can produce estimates of posterior marginal probabilities of the type indicated through examples in the Significance Statement. Note that these marginal probabilities, that are represented by the firing rates of corresponding neurons in $\mathcal{N}$, integrate automatically information from many modes of the learnt approximation $p(\mathbf{y}; \boldsymbol{\theta})$ to p^*. Hence if these modes represent individual memory items of a memory model, the network $\mathcal{N}$ can combine information from many different memory items (episodes), also in ways that could not be anticipated during learning.

Small numbers of hidden neurons in the learning modules often suffice

The structure of the network $\mathcal{N}$ in Fig. 5 is very similar to the structure of a constructed network of spiking neurons that directly mimics a representation of p^* by a Bayesian network according to (Pecevski et al., 2011). But there the number of hidden neurons $\boldsymbol{\alpha}^k$ for a random variable y^k was required to be exponentially large in the number of variables in the Markov blanket of y^k. In contrast, in the learning

approach of this article, one can employ in principle any number, also a very small number, of hidden neurons in $\boldsymbol{\alpha}^k$. The described learning approach will approximate the marginal distribution of p^* over y^k and the Markov blanket of y^k with a mixture distribution whose number of modes is determined by the chosen number of hidden neurons in $\boldsymbol{\alpha}^k$. For example, in Example 1 we had chosen just two hidden neurons in $\boldsymbol{\alpha}$ for each of the two possible values of z, instead of two times four that were used in the construction of (Pecevski et al., 2011) for representing all four possible assignments of values to the inputs $\mathbf{x}$ of a module. But as the comparison of panels B and C (and of D and E) in Fig. 3 shows, the network with the smaller number of hidden neurons $\boldsymbol{\alpha}$ works in this case about as well as the larger network. This effect is predicted from general results in learning theory: A learning network with fewer parameters sacrifices representation power, but gains generalization capability. Furthermore, naturally occurring distributions can often be approximated quite well by mixture distributions with a relatively small number of components (modes). This suggests that real world distributions p^* of examples can often be learnt by relatively small networks $\mathcal{N}$. An ideal scenario from a biological perspective would be one where a population of hidden neurons in $\mathcal{N}$ can become larger if the size provides insufficient resolution or prediction capability for the examples that it receives.

Note that with this approach a spiking network $\mathcal{N}$ can in principle learn an approximation of a given distribution p^* even without prior information on the dependency structure among RVs of p^*. One can set up the network $\mathcal{N}$ so that each module $\mathcal{N}^k$ extracts the probabilistic association between RV y^k and *all* other RVs $\mathbf{y}^{\setminus k}$ (i.e., replacing $\mathbf{y}^{B(k)}$ by all RVs other than y^k). The size (i.e., number of hidden neurons $\boldsymbol{\alpha}$, see Fig. 2) of $\mathcal{N}^k$ determines the quality of the resulting learnt approximation of $p^*(y^k|\mathbf{y}^{\setminus k})$. Whereas a good approximation can only be theoretically guaranteed if the number of hidden neurons in $\mathcal{N}^k$ is exponential in the number K of RVs of p^*, acceptable results may emerge with drastically fewer hidden neurons, provided that their number is in the same range as the sum of the number of main modes of the distributions $p^*(\mathbf{y}^{\setminus k}|y^k = l)$ for different values l (see Fig. 3 for an illustration).

Example 2: Autonomous learning of explaining away in perceptual inference

We demonstrate learning in recursive combinations of the basic learning module for a concrete example with 4 modules (see Fig. 6C). We apply it to the task of learning a complex distribution p^* that represents a standard example for explaining away in visual perception. The famous experiment of (Knill and Kersten, 1991), depicted in Fig. 6, had first demonstrated that nontrivial inference is involved in visual perception. A subsequent study (Kersten and Yuille, 2003) proposed that this perceptual effect can be understood as "explaining away" in probabilistic inference, and a Bayesian network with 4 RVs, $y^1, \ldots, y^4$ (see Fig. 6B) was introduced to demonstrate this. The probability distribution of the Bayesian network is $p(y^1, y^2, y^3, y^4) = p(y^1)p(y^2)p(y^3|y^1, y^2)p(y^4|y^2)$, and the inference task is to calculate the marginal posterior probability distributions $p(y^1|y^3, y^4)$ and $p(y^2|y^3, y^4)$.

In the experiment of (Knill and Kersten, 1991), the demonstrated different perception of the shading in the two surfaces in Fig. 6A (see caption of Fig. 6A for more details) can be explained by two different competing causes, either by different relative reflectance of the two abutting surfaces (y^1), or by their cylindrical 3D shape (y^2). An observed curved contour of the surfaces is a cue that increases the probability of a cylindrical 3D shape. Since a cylindrical 3D shape alone is enough to explain the shading, it reduces the probability that the relative reflectance is different. Hence, one of the competing causes, the cylindrical 3D shape, "explains away" the other possible cause, the different reflectance of the surfaces. In the other case, when a flat contour is observed as an additional cue, this increases the probability of a rectangular 3D shape. As a rectangular 3D shape can not explain the observed shading, the probability of the second possible cause for the shading, the different reflectance is increased. This type of "explaining away" in probabilistic inference can only occur for distributions p^* that have higher-order interactions between 3 or more RVs, like between the two competing causes and the observed shading in this example.

We show that the underlying distribution p^* can be learnt (approximately) from examples for this visual perception task, and that the network $\mathcal{N}$ which learns this approximation learns simultaneously to deal with the explaining away effect as an emergent phenomenon.

The structure of the neural network $\mathcal{N}$ suitable for learning this target probability distribution p^*

is given in Fig. 6C. It consists of four interconnected learning modules, where the connections between the learning modules reflect the dependencies between the RVs in the Bayesian network in Fig. 6B. Additionally, the structure of one of the learning modules, the learning module $\mathcal{N}^1$ for the RV y^1, is given in Fig. 6D in detail. Each subgroup $\boldsymbol{\alpha}^{1l}$ of hidden neurons has 2 neurons. This number of hidden neurons is smaller than the number of hidden neurons in the exact neural implementations of this Bayesian network in (Pecevski et al., 2011) (Implementation 2), equal to the total number of assignments of values to the RVs in the Markov blanket, which in this case is four. But we show that the smaller neural network can nevertheless learn the distributions $p^*(y^2, y^3|y^1 = l)$, since these distributions do not have more than 2 modes. In fact, we use here just two hidden neurons in the subgroups $\boldsymbol{\alpha}^{kl}$ of all learning modules, also for the learning module $\mathcal{N}^3$ where the total number of assignments of values to the RVs in the Markov blanket is 8. As we will see in the results, two hidden neurons in the learning module $\mathcal{N}^3$ are enough to learn a good approximation of $p^*(y^1, y^2, y^4|y^3 = l)$.

We performed computer simulations of learning with this network, where examples drawn from the target probability distribution p^* were presented to the network successively during learning. The distribution p^* was defined according to Table 4 in Methods in order to capture the visual perception scenario of Fig. 6 in a qualitative manner. The learning phase took 1200s of simulated biological time, and each example was presented for a time period of 100ms. The weights and biases of the neurons were randomly initialized before learning.

We first analyzed the stationary distribution of network states in the network $\mathcal{N}$ from Fig. 6C after learning. Fig. 7A shows that the network switches spontaneously between different network states, and occasionally remains longer in one of the network states that have high probability under the stationary distribution (see Fig. 6B). One can relate the firing activity of this network $\mathcal{N}$ to an approximation $p(\mathbf{y}; \boldsymbol{\theta})$ of the target distribution $p^*(\mathbf{y})$ by assuming in the usual manner (as e.g. in (Berkes et al., 2011) and (Buesing et al., 2011)) that the firing of a neuron in the population code for variable y^k sets the value of this variable for a time period of length τ to the value encoded by this neuron. The distribution over 4 binary random variables $y^1, \ldots, y^4$ obtained in this way from the spontaneous firing of the network $\mathcal{N}$ is shown in Fig. 7B and compared to the target distribution p^*.

In Fig. 8A we examined how the learning in the modules progresses in time. In each of the 4 learning modules in the network (one for each problem variable y^k, see Fig. 5 and Fig. 6C) the Kullback-Leibler (KL) divergence between the marginal target distribution $p^*(y^k, \mathbf{y}^k)$ that it learns to approximate and its internal model $p^k(y^k, \mathbf{y}^{B(k)}; \boldsymbol{\theta})$ decreases and stabilizes to a local minimum after about 300 seconds. The same is true for the sum of all KL divergences of the modules (see Fig. 8B). As a result, the difference between the model distribution $p(\mathbf{y}; \boldsymbol{\theta})$ of the network and the full target distribution $p^*(\mathbf{y})$ also decreases, as shown in Fig. 8C. This is because after learning a good internal model of the marginal target distributions $p^*(y^k, \mathbf{y}^k)$, the firing probability of the output neurons of the learning modules approximate well the conditionals $p^*(y^k|\mathbf{y}^{B(k)})$ of p^*, as shown in Fig. 8D, which according to the theory leads to a good approximation of p^* by the network.

The PSPs and STDP curves in the simulated network have a more biological smoother alpha shape, which differs from the rectangular shape used in the theory. This can introduce minor deviations of the learning convergence from the theoretically optimal one, as for example the slight increase of the KL divergence in Fig. 8C in the second half of the learning process.

This network $\mathcal{N}$ can extract the information that it has acquired from the examples in a very flexible manner through probabilistic inference. For example, if evidence $\mathbf{e}$ is entered for some of the problem variables $y^1, \ldots, y^4$ (by inducing corresponding neurons in their population codes to fire at high rates) the conditional marginal probabilities of other variables y^i can be read off from the firing rates of neurons that represent y^i through population coding. In particular, we demonstrate in Fig. 9 that the network $\mathcal{N}$ has acquired autonomously from examples the capability to carry out non-trivial probabilistic inference that involves explaining away (i.e., higher order dependencies among random variables).

Finally, we would like to point out that our approach is not restricted to Bayesian networks, it can be applied to any type of graphical models, e.g. also to Markov random fields and factor graphs. The network connectivity is determined by the Markov blankets of the random variables which can easily be read out from the graph structure of any graphical model.

Discussion

Numerous models for probabilistic inference in networks of neurons have been proposed, and many models for the impact of learning on network computations have been proposed. But surprisingly, these two lines of research have so far (with a few exceptions that are discussed below) not been brought together. We propose in this article a new model for learning and memory organization in recurrent networks of spiking neurons that makes the information that is gathered from numerous experiences immediately available for complex and unforeseen memory retrievals in the form of probabilistic inference. The network is able to carry out such probabilistic inference just through its inherent stochastic dynamics. More precisely, we assume that the network receives samples (=examples) from some unknown multivariate distribution p^*, and show that a suitably structured recurrent network of spiking neurons is able to build through spike-timing-dependent plasticity (STDP) and intrinsic plasticity of the excitability of neurons an internal model for p^* in such a way, that it can answer a diverse repertoire of probabilistic inference queries about the stored knowledge through sampling. Early models for memory storage and retrieval in recurrent networks of artificial neurons (Steinbuch, 1961; Hopfield, 1982; Hopfield, 1984) had focused on the storage of isolated memory items in attractors of a deterministic network dynamics. The only memory queries which are considered in these models are input completion tasks, or finding the most similar memory item to the input pattern. Answering queries that require the combination of several stored memory items is virtually impossible in such model, especially since the stored memory items were required to be scrambled (orthogonalized) before they were committed to the network. This is inconsistent with experimental data on human memory, which require a substantially more structured memory organization (Stickgold and Walker, 2013).

We have shown in this article, that very simple network motifs (see Fig. 2) provide learning modules, that can extract probabilistic relationships between random variables from examples, simply by applying STDP and intrinsic plasticity. One feature of these network motifs is that different groups of neurons on layer 2 project to different neurons on layer 3. Such non-convergent synaptic connections are difficult to identify with current experimental methods, but have already been found in the mouse cortex (Chen et al., 2013), where largely non-overlapping populations of pyramidal cells in layer 2/3 of area S1 project to areas S2 and M1. Somewhat similar fine-scale connectivity patterns had previously been found within a cortical column of rat visual cortex (Yoshimura et al., 2005; Kampa et al., 2006). It remains to be tested whether these fine-scale network structures in the brain support the learning of stochastic associations as predicted by our model.

These simple stochastic association modules can be recursively combined (see Fig. 4, 5), and are then able to learn also complex stochastic relationships, including higher order moments (e.g. explaining away), from examples. We have demonstrated this for a well-known visual inference task from (Knill and Kersten, 1991) that is known to require explaining away (Fig. 6 - 9), but where it has been an open question whether brains could in principle acquire this capability through learning. Furthermore we have shown that this impressive learning capability of recurrent networks of spiking neurons can be understood for simple rules for STDP and intrinsic plasticity of neurons on the basis of a rigorous mathematical theory (Expectation Maximization).

The representations of statistical knowledge in networks of spiking neurons that are shown here to result from learning are structurally very similar to previously proposed ones in (Pecevski et al., 2011) that were based on construction instead of learning, with one important difference: Whereas the constructed networks required very large numbers of auxiliary neurons for representing random variables with larger Markov blankets, we show here that similar networks but with realistic numbers of hidden (auxiliary) neurons can still provide good approximations. These smaller modules automatically learn to make optimal use of the number of auxiliary neurons that are available to them. They learn automatically to approximate complex stochastic associations between random variables that emerge from examples, with a mixture distribution whose maximal number of components (modes) fits to the available number of auxiliary neurons. We have demonstrated this important feature of our learning approach in Fig. 3 (compare panels B,D with C,E) and in our model for explaining away in Fig. 6 (where the number of auxiliary neurons in the learning modules, see panel D, is smaller than the number required by the

construction of (Pecevski et al., 2011)).

One other interesting general feature of our model is that it shows how noise in networks of neurons can be beneficial. In fact, noise provides here a necessary ingredient both for the self-organization of the network through STDP, and for the use of learnt probabilistic relationships for probabilistic inference via MCMC sampling (Habenschuss et al., 2013a). The necessary stochasticity in the neurons can come from different sources like for example the unreliable neurotransmitter release from the vesicles at the presynaptic site, or the stochastic closing and opening of membrane ion channels (Faisal et al., 2008).

Finally, we would like to point out that our network model produces answers to probabilistic inference queries (see e.g. Fig. 9) in the form of firing rates, which is obviously useful for communicating such answers to downstream networks. Internally however, the network works with a spike-based encoding of network states, where every spike has an impact on the network state (see e.g. Fig. 7).

Our model and theory has identified concrete plasticity mechanisms and network architectures that would enable networks of neurons in the brain to build probabilistic internal models for their stochastic environment. We have focused here on an idealized model in order to keep it theoretically tractable. Further work will have to explore to what extent similar learning phenomena arise in more complex neural network models that sacrifice theoretical tractability for additional biological details.

Related work

There are several studies that propose neural implementations of probabilistic inference in general graphical models where, as in our approach, the present independencies of the distribution are directly exploited for reducing the complexity of the neural network structure. The majority of these base their implementations on the loopy belief propagation algorithm (Rao, 2006; Steimer et al., 2009; Siegelmann and Holzman, 2010; Litvak and Ullman, 2009). Except for (Siegelmann and Holzman, 2010), to the best of our knowledge, none of these studies proposes a way how these neural structures could emerge through learning from examples. The model of (Siegelmann and Holzman, 2010) is formulated on a more abstract level. It is based on the observation that belief propagation (message passing) requires only three arithmetical operations: summation, multiplication, and division (for normalization). Their network model is based on symbolic computational units (interpreted as multi-compartment neurons) that carry out these arithmetical operations on real numbers. The resulting real numbers are interpreted as probabilities or messages that are sent to other units during belief propagation. Estimates of conditional probability tables are extracted from examples through online accumulators, assumed to be implemented as plasticity of the weight of a dendritic branch that represents a specific value assignment for a set of random variables (more precisely, for the Markov blanket of a random variable). Hence this learning model is not based on synaptic plasticity, but rather on a plasticity mechanism that changes the weight of a whole dendritic branch. We are not aware of an attempt to implement this approach with spiking neurons, or with more local plasticity rules.

An alternative framework for probabilistic inference in neural circuits developed in (Ma et al., 2006; Beck et al., 2008; Beck et al., 2011; Beck et al., 2012) is based on representation of probability distributions in probabilistic population codes. To the best of our knowledge, the question of how the neural implementations in those studies can emerge through learning with local plasticity mechanisms has so far not been addressed.

We have focused in this paper on the task of learning a time-invariant distributions p^* over static patterns. Complementary to this, in several studies (Deneve, 2007; Kappel et al., 2014; Rezende et al., 2011; Brea et al., 2013) the authors developed neural network models for learning time-varying distributions, restricted to dynamical Bayesian networks that do not have dependencies between the RVs in the same time step, which simplifies the learning. In these models the network learns to reproduce sequences of patterns, by developing latent representations as a memory about the recent history of patterns, and learning the stochastic transitions between the patterns in the sequence. In contrast to this, the neural networks in our approach learn a probability distribution of static patterns as their stationary distribution, where the distribution can contain arbitrary dependencies between the random variables without any restrictions.

The problem of learning a probability distribution from examples has been well studied in the artificial neural network community. The Boltzmann machine is one of the earliest developed neural networks that can learn general probability distributions (Ackley et al., 1985). The learning in the Boltzmann machine is however difficult for learning higher-dimensional probability distributions with a larger number of RVs (Hinton, 2002; Carreira-Perpinan and Hinton, 2005). Building on the Boltzmann machine idea, more recently developed deep belief networks have considerably improved the efficiency and scalability of learning by using a Boltzmann machine with a two-layer bipartite graphical model structure, called restricted Boltzmann machine, which is easier to train than the general Boltzmann machine (Hinton et al., 2006). Deep belief networks learn multi-layer latent representations of the input data in a generative model by training layer-by-layer restricted Boltzmann machines. As in our approach, the learned probability distribution is embodied as a stationary distribution of the network states. However, the structure of Boltzmann machines and deep belief networks exhibits symmetric weights between the neurons which are not found in biological networks of neurons, and also the biological plausibility of layerwise training and the learning rules in these networks is not clear.

These approaches derive their learning rules from the maximum likelihood learning principle. Our learning approach for network of interconnected learning modules, on the other hand, is more reminiscent to maximizing the pseudo-likelihood (Besag, 1975), an alternative parameter estimation method in statistical learning, where the objective function can be formulated through the Kullback-Leibler divergences between the target and the model conditional distributions of one variable given the rest of the RVs. The difference is that in the pseudo-likelihood objective function the conditional distributions are all derived from a single probabilistic model, whereas in our approach each learning module learns a separate generative model of a marginal of the target probability distribution.

Experimentally testable predictions of our model and other questions for further research

Our model makes significantly different predictions compared with previous models for memory in neural networks with regard to the architecture of the neural networks involved. Models based on Hopfield networks and Boltzmann machines predict that memories are stored in homogeneous networks of only excitatory neurons with symmetric synaptic connections. In contrast, our model predicts that memories are stored in large assemblies of highly structured generic microcircuit motifs, each consisting of excitatory and inhibitory neurons. Our model gives rise to a concrete hypothesis, why different species have different learning capabilities, that cannot be explained in terms of the different numbers of neurons in their brains. It proposes, that the structure or structural predisposition of interconnections between neurons is an important factor for learning performance. In particular, it predicts that the superior learning capabilities result from a genetically more precisely structured interaction of excitatory and inhibitory neurons. Our model of a probabilistic learning module (see Fig. 2) proposes in fact two different functional roles of inhibitory neurons: Lateral inhibition among excitatory neurons that induces each of them to specialize on different presynaptic firing patterns, and another type of inhibition that prevents hidden neurons that learn a probabilistic relationship between random variables $\langle \mathbf{x}, z \rangle$ for a specific value $z = l$, to engage plasticity for examples $\langle \mathbf{x}, z \rangle$ with $z \neq l$. This architecture provides concrete hypotheses for the analysis of data on the role of inhibitory neurons in the organization of plasticity (Caroni, 2015; Letzkus et al., 2015).

Furthermore our model for an atomic probabilistic learning module predicts that pyramidal cells within a network can represent probabilistic, rather than only deterministic relationships. This can in principle be tested experimentally by exciting (e.g. through optogenetic methods) a subset A of these neurons, and observe the resulting firing probability of pyramidal cells B. Furthermore our model predicts that this firing probability of neurons in B can be changed upwards through trials where both neurons in A and B are made to fire, and changed downwards through trials where neurons in A are made to fire and neurons in B are inhibited.

In contrast to preceding memory models, our model does not predict that synaptic connections between neurons are in general symmetric. This would actually be impossible for synaptic connections

between excitatory and inhibitory neurons. But also for synaptic connections between pyramidal cells in the cortex a symmetry of synaptic weights between them is not really consistent with the currently available experimental data (see e.g. (Song et al., 2005) and Fig. 4 in (Haeusler et al., 2009)). With regard to the plasticity mechanisms involved, our model points to an essential contribution of intrinsic plasticity of the excitability of pyramidal cells for memory formation (Mozzachiodi and Byrne, 2010).

An interesting functional prediction of our model is that memory recall can not only take the form of input completion (like in a Hopfield network), but can engage the full power of probabilistic inference. Furthermore it proposes that this inference is implemented through the inherent stochastic dynamics of networks of neurons in the brain. Such implementation has previously already been proposed on the basis of data from cognitive science, see e.g. (Denison et al., 2013) and (Vul and Pashler, 2008). Furthermore our model predicts that memory recall and imagination of possible scenarios are closely related brain computations, that engage similar network mechanisms. This prediction appears to be consistent with recent experimental data, which suggest that memory recall and imagination/fabulation engage largely the same brain systems (Schacter, 2002).

Finally, our model predicts that the development of suitable brain networks that store basic insights about typical causal roles and dependencies of the objects and phenomena we encounter is essential for learning. This prediction is in line with experimental results from cognitive science, which argue that such basic knowledge about dependencies in the real world is already known to 2-year old children (Landau et al., 1988). Since our learning approach emphasizes the role of probabilistic inference, it also provides a theoretical framework for integrating innate or previously learnt knowledge in the form of priors.

The learning paradigm that we have presented was designed to provide an alternative to other neural network models for higher level memory. Such higher level memory system in the brain receives high-dimensional inputs $\mathbf{y}$ from numerous brain areas, in particular also higher level features that are extracted from sensory inputs by other learning systems.

It is at this point an open question to what extent the proposed model for learning in networks of spiking neurons can also provide insight into the organization of lower level learning systems in the brain, for example in the visual system. The underlying mathematical approach is quite general, and guarantees convergence to an internal model for any external distribution p^* over discrete RVs that generates the examples $\mathbf{y}$ that are presented to the network. How good this internal model will become is related to the question how well p^* can be approximated by a Bayesian network with small degrees of nodes, or more generally, by a simplified distribution where all RVs have small Markov blankets. However a precise answer to this question is even more difficult, since our model is in principle also able to approximate some distributions with large Markov blankets, see the remarks at the end of the subsection "Small numbers of hidden neurons in the learning modules often suffice" in Results. Numerical tests for a number of practically relevant distributions p^* are likely to provide further insight into this question.

In principle it is also possible to boost the learning capability of our approach by stacking multiple copies of the learning network. The $\boldsymbol{\alpha}$ neurons in the network learn to encode salient combinations of values of different RVs, similarly as feature detectors on the first hidden layer of a deep learning network. Hence it would make sense to send the output of these $\boldsymbol{\alpha}$ neurons also as input to a version of the same type of learning network on a second level. One would then expect that the second level network learns in the same unsupervised manner to detect and represent salient combinations of values in these $\boldsymbol{\alpha}$ neurons. The analysis of the performance of such a stacked learning architecture based on spiking neurons and STDP is a topic for future research.

Summary

Altogether we have shown that some forms of probabilistic inference can be learnt through STDP, even in cases where the nontrivial "explaining away" effect occurs, see Fig. 6-9. We propose, that this new paradigm for network learning provides an alternative to previous models for associative learning that were based on learning categorical rather than probabilistic associations. In addition, in contrast to earlier memory models, this new model is compatible with basic properties of biological networks of neurons, such as spikes, trial-to-trial variability, stereotypical microcircuit motifs, and STDP.

Methods

We first present a rigorous learning theory that supports the learning results that are presented. After that, we provide details to the computer simulations.

Theoretical properties of the basic learning module (stochastic association module) and its plasticity

In this section we give additional details about the structure of the module, the firing of the output neurons and the learning procedure. In particular, the subsequent subsection titled "Implicitly represented generative model" defines the probabilistic model $p(\mathbf{x}, z; \boldsymbol{\theta})$ represented in the module. After that, in the two subsections "Firing probability of the output neurons resulting from the internal generative model" and "Deriving the probability distribution of the firing of the neurons $\boldsymbol{\alpha}$" the firing of the output and the alpha neurons in the module is expressed through the represented probabilistic model. The last subsection "The plasticity rules minimize the KL divergence through Expectation Maximization" continues with the explanation how the plasticity rules implement learning of the stochastic associations between the variables $\mathbf{x}$ and z through EM.

The neurons $\boldsymbol{\alpha}$ in the learning module are interconnected through inhibitory inter-neurons (not shown in Fig. 2) that enforce strong lateral inhibition between them. The role of the lateral inhibition is to ensure that if one of the $\boldsymbol{\alpha}$ neurons fires at time t, no other neuron in $\boldsymbol{\alpha}$ will fire within the interval $(t, t+\tau)$. Each neuron in population $\boldsymbol{\alpha}$ receives input synaptic connections from all input neurons $\boldsymbol{\chi}$ such that the firing of the population $\boldsymbol{\chi}^i$ encodes the value of the input variable x^i (see Fig. 2). For each non-zero value of the input variable $x^i = m$ there is a dedicated neuron $\boldsymbol{\chi}^{im}$, and if $\boldsymbol{\chi}^{im}$ fired in the time period $(t-\tau, t]$, the value of x^i at time t is $x^i = m$. The firing in the population $\boldsymbol{\chi}^i$ is such that no two neurons fire in the time interval $(t-\tau, t]$, i.e. after a spike there is no spiking period of duration τ. If there is no spike in the interval $(t-\tau, t]$, then the value of x^i is 0.

All neurons in the population $\boldsymbol{\alpha}^l$ connect to their corresponding output neuron ζ^l. These synaptic connections are not subject to synaptic plasticity and have a fixed strong efficacy. The strong weights of these synapses achieve that whenever any of the neurons in $\boldsymbol{\alpha}^l$ fires, it drives the output neuron ζ^l to fire. The output neurons fire only if they receive input spikes, otherwise they remain silent. Thus, the spikes that the neuron ζ^l emits can be seen as a union of all spikes of the hidden neurons in the group $\boldsymbol{\alpha}^l$.

One can give an analytical description for the firing probability density of the output neuron ζ^l according to the model. In order to do that, first we need to introduce some notation. With $w^l_{im,j}$ we denote the efficacy of the synaptic connection from the input neuron χ^{im} to the neuron α^l_j from the group $\boldsymbol{\alpha}^l$. For the bias (intrinsic excitability) of the neuron α^l_j we write b^l_j. We also introduce, for simplicity, binary RVs x^{im} corresponding to each neuron χ^{im}. The binary RV x^{im} assumes value 1 at time t if the current value of the RV x^i is equal to m, and 0 otherwise. The current value of x^{im} at time t can be determined just from the spikes of the neuron χ^{im}, i.e. it is 1 if χ^{im} fired within the time interval $[t, t-\tau]$, and 0 otherwise. With this notation, we can describe the firing probability density of ζ^l at time t by

$$\rho^l(t) = \sum_{j=1}^{J^l} \frac{1}{\tau} \exp(u^l_j(t)) = \sum_{j=1}^{J^l} \frac{1}{\tau} \exp(b^l_j + \sum_{i=1}^{I} \sum_{m=1}^{M(x^i)} w^l_{im,j}\, x^{im}(t)) \qquad , \tag{4}$$

where u^l_j is the membrane potential of neuron α^l_j, J^l is the number of neurons in the subpopulation $\boldsymbol{\alpha}^l$, and $M(x^i)$ denotes the maximum integer value that x^i can assume. The firing probability density of the output neuron ζ^l is equal to the sum of the firing probability densities of all neurons in the subpopulation $\boldsymbol{\alpha}^l$, which follows from the specific connectivity structure in the learning module. The sums with the indices i and m iterate over all input neurons χ^{im}.

During learning we assume that examples $\langle \tilde{z}(0), \tilde{\mathbf{x}}(0) \rangle$, $\langle \tilde{z}(1), \tilde{\mathbf{x}}(1) \rangle$, ..., $\langle \tilde{z}(n), \tilde{\mathbf{x}}(n) \rangle$, ... drawn from the joint probability distribution $p^*(z, \mathbf{x})$ are presented to the WTA circuit in succession, one by one. In an example the variable z has an integer value from the set $\{1, \ldots, M(z)\}$ ($M(z)$ denotes the maximum integer value that z can assume), and each of the variables x^i assumes a value from the set $\{1, \ldots, M(x^i)\}$. Each example $\langle \tilde{z}(n), \tilde{\mathbf{x}}(n) \rangle$ is presented for several tens or hundreds of milliseconds. During this time period, the input neurons fire according to the values of the input variables $\tilde{\mathbf{x}}(n)$. In particular if $\tilde{x}^i(n) = m$, then the input neuron χ^{im} fires with a high firing rate, whereas all other input neurons in the population $\boldsymbol{\chi}^i$ are silent. The value $\tilde{z}(n)$ in the example is given to the network via inhibitory currents in a subset of the neurons $\boldsymbol{\alpha}$. More precisely, if in the current example $\tilde{z}(n) = l$, then all neurons in $\boldsymbol{\alpha}$ that do not belong to $\boldsymbol{\alpha}^l$ are inhibited with a strong negative external current which prevents them from firing. At the same time the neurons in the group $\boldsymbol{\alpha}^l$ do not receive any external inhibitory currents and are free to fire according to their firing probabilities determined by their inputs. At the beginning of the learning process it is assumed that the biases of the $\boldsymbol{\alpha}$ neurons have large values so that they fire with high firing rates regardless of what presented input $\tilde{\mathbf{x}}(n)$ is.

As stated above, in the presented examples during learning the variables z and x^i $(i = 1, \ldots, I)$ assume values from the sets $\{1, \ldots, M(z)\}$ and $\{1, \ldots, M(x^i)\}$ respectively. These values could represent states of the external environment, internal beliefs or any other behavior related variable value. However, in the population coding of values of the variables z and x^i $(i = 1, \ldots, I)$ by their corresponding neuron populations in the learning module, the variables can have an additional value of zero. A neuron population that encodes a variable through population coding assigns value zero to this variable if none of the neurons fires for a period longer than τ. There is not a dedicated neuron whose firing signals a zero value, as is the case for the rest of the non-zero values of the variable. The assumed spike based encoding with a zero value is the same as in the neural sampling theory (Pecevski et al., 2011; Buesing et al., 2011). It defines a value for the variable at each moment in the continuous time dynamics of the neural network model.

The zero values in the population coding of the learning module do not represent states of the external environment. The reason for this is that learning module does not learn the probabilities $p(\mathbf{x}, z; \boldsymbol{\theta})$ in its internal model where some of the variables z and x^i $(i = 1, \ldots, I)$ have zero value. These probabilities have very fixed small (close to zero) values (see subsection "Firing probability of the output neurons resulting from the internal generative model" of this section of Methods). Therefore, it is assumed that the examples contain only the encoded variable values whose stochastic relations can be learnt by the learning module.

Implicitly represented generative model. The learning theory of the module is based on its internally represented generative model $p(z, \mathbf{x}; \boldsymbol{\theta})$, as it is discussed in section "A network module for learning stochastic associations" of Results (see also section "The plasticity rules minimize the KL divergence through Expectation Maximization" of Methods). In order to define the implicitly represented generative model, we first define an additional compound RV $\boldsymbol{a}$ in form of a vector of binary RVs, that we will relate to the stochastic activity of the neurons in the population $\boldsymbol{\alpha}$. More precisely, the RV $\boldsymbol{a}$ is defined as a population code, represented as a set of binary RVs a^l_j, one for each neuron α^l_j. The generated values of a^l_j over time during the activity of the WTA circuit are defined by the spiking of α^l_j in the same way as the spiking of the input neurons $\boldsymbol{\chi}$ define the values of $\mathbf{x}$. The value of a^0_0, which does not have a corresponding neuron in the population $\boldsymbol{\alpha}$, is defined as $a^0_0 = 1 - \sum_{l,j} a^l_j$. Note that as there can be no two neurons in $\boldsymbol{\alpha}$ spiking within the time window $(t - \tau, t)$ because of the lateral inhibition, it follows that the vector RV $\boldsymbol{a}$ has a restricted domain of allowed values, consisting only of values where exactly one a^l_j is equal to 1, and the others are 0. The value of $\boldsymbol{a}$ where only $a^l_j = 1$ and the rest are 0 will be denoted by $\boldsymbol{a} = (l, j)$.

Having defined $\boldsymbol{a}$ we can now write the parametrized form of the generative model as follows

$$p(\boldsymbol{a}, z, \mathbf{x}; \boldsymbol{\theta}) = \frac{1}{A(\boldsymbol{\theta})}\, p(z|\boldsymbol{a}) \exp\left(\sum_{i,m} \sum_{l,j} \hat{w}^l_{im,j}\, x^{im}\, a^l_j + \sum_{l,j} \hat{b}^l_j a^l_j \right) \quad , \tag{5}$$

where $A(\boldsymbol{\theta})$ is the normalization constant. In the sum over the indices i and m, the index i iterates through all input variables from x^1 to x^I. For each value of the index i, the index m iterates from 0 to $M(x^i)$, i.e. through the set of possible values of the RV x^i. In the two sums with the indices l and j, l iterates through all possible values of z from 0 to L, and j iterates from 1 to J^l (Note that $J^0 = 1$ as there is not a group of neurons in $\boldsymbol{\alpha}$ for $l = 0$, and hence only one parameter). The parameter vector $\boldsymbol{\theta}$ consists of all parameters $\hat{w}^l_{im,j}$ and $\hat{b}^l_j$ which are encoded in the synaptic weights and biases of the alpha neurons. After learning, the marginal distribution $p(z, \mathbf{x}; \boldsymbol{\theta})$ of (5) becomes an internal model of the distribution $p^*(\mathbf{x}, z)$ of the presented examples.

The probability $p(z|\boldsymbol{a})$ in (5) is not parametrized and specifies the deterministic relations between $\boldsymbol{a}$ and z. It is defined as follows: $p(z = l_1|\boldsymbol{a} = (l_2, j)) = 1$ if $l_1 = l_2$, and $= 0$ otherwise, for all $l_1 \in \{0, \ldots, L\}$, and all (l_2, j) that are valid values of $\boldsymbol{a}$. These probability values express the fact that when the neuron α^l_j fires and sets the value of $\boldsymbol{a}$ to be (l, j), then this also uniquely determines z, i.e sets the value of z to $z = l$.

A sufficient condition for having a normalization constant $A(\boldsymbol{\theta}) = 1$ are the following constraints on the parameters

$$\begin{aligned} &\sum_m \exp(\hat{w}^l_{im,j}) = 1 \quad \text{for all } i \in \{1, \ldots, I\},\, j \in \{1, \ldots, J^l\} \text{ and } l \in \{0, \ldots, L\}, \text{ and} \\ &\sum_l \sum_j \exp(\hat{b}^l_j) = 1 \quad . \end{aligned} \tag{6}$$

As we will see later, one property of the plasticity rules is that they move the parameter vector towards the region of the parameter space where these normalization constraints are approximately satisfied, i.e. they try to keep the internal probabilistic model normalized.

If one assumes that the normalization constant satisfies $A(\boldsymbol{\theta}) = 1$, then the generative model in (5) obtains the form

$$p(\boldsymbol{a}, z, \mathbf{x}; \boldsymbol{\theta}) = p(\boldsymbol{a}; \boldsymbol{\theta}) p(z|\boldsymbol{a}) \prod_{i=1}^{I} p(x^i|\boldsymbol{a}; \boldsymbol{\theta}) \quad , \tag{7}$$

By marginalizing $\boldsymbol{a}$ in the full generative model $p(\boldsymbol{a}, z, \mathbf{x}; \boldsymbol{\theta})$, we obtain the marginal probability distribution $p(z, \mathbf{x}; \boldsymbol{\theta})$ which has the form of a mixture of multinomials. The marginal probabilistic model $p(z, \mathbf{x}; \boldsymbol{\theta})$ models the probabilistic relations between the RV z, encoded by the firing of the output neurons, and the RVs $\mathbf{x}$, which are encoded in the inputs. These relations are modeled through the vector of auxiliary RVs $\boldsymbol{a}$ which are also called hidden RVs. In the mixture model the conditional probabilities $p(z|\boldsymbol{a})$ and $p(x^i|\boldsymbol{a}; \boldsymbol{\theta})$ for $i = 1, \ldots, I$ are the likelihoods, and $p(\boldsymbol{a}; \boldsymbol{\theta})$ is the prior.

It can be easily shown from (5) that by assuming $A(\boldsymbol{\theta}) = 1$ the likelihoods $p(x^i|\boldsymbol{a}; \boldsymbol{\theta})$ for $i = 1, \ldots, I$ are

$$p(x^i = m|\boldsymbol{a} = (l, j); \boldsymbol{\theta}) = \exp(\hat{w}^l_{im,j}) \qquad \text{for all } l,\, j \text{ and } m \quad . \tag{8}$$

Similarly the priors are

$$p(\boldsymbol{a} = (l, j); \boldsymbol{\theta}) = \exp(\hat{b}^l_j) \qquad \text{for all } l,\, j \quad . \tag{9}$$

Hence, given that the normalization constraints (6) hold, the parameters $\hat{w}^l_{im,j}$ are equal to the log of the conditional probabilities from the likelihood $\hat{w}^l_{im,j} = \log p(x^{im} = 1 | a^l_j = 1; \boldsymbol{\theta})$, whereas $\hat{b}^l_j$ represent the log probabilities of the prior $\hat{b}^l_j = \log p(a^l_j = 1; \boldsymbol{\theta})$.

For convenience, Table 1 lists the mathematical notation that is used in the definition of the learning module.

Firing probability of the output neurons resulting from the internal generative model. Here we will express the firing probability density of the output neurons through the probabilistic model $p(z, \mathbf{x}; \boldsymbol{\theta})$. This will show that by learning the internal model $p(z, \mathbf{x}; \boldsymbol{\theta})$ of the target distribution $p^*(z, \mathbf{x})$, the output neurons actually learn to fire according to the conditional $p^*(z|\mathbf{x})$. Thus, as discussed in section "A network module for learning stochastic associations" of Results, the module exhibits the learned stochastic associations between the variables z and $\mathbf{x}$ through the firing of its output neurons given input $\mathbf{x}$. Furthermore, this particular form of firing enables to recurrently interconnect multiple modules in larger networks to learn more complex distributions. Networks of interconnected learning modules are presented in section "Recursive combinations of the basic learning module enable efficient learning of complex distributions from examples" of Results and further discussed in section "Theoretical properties of networks of recursively interconnected basic learning modules" of Methods.

As a prerequisite to the derivation, we first establish a relation between the parameters $\hat{w}^l_{im,j}$ and $\hat{b}^l_j$ of the probabilistic model $p(\boldsymbol{\alpha}, z, \mathbf{x}; \boldsymbol{\theta})$, and the synaptic weights $w^l_{im,j}$ and biases b^l_j of the neurons in $\boldsymbol{\alpha}$. For the synaptic weights $w^l_{im,j}$ we assume that they are equal to the corresponding parameters $\hat{w}^l_{im,j}$ shifted by a constant baseline value $w^l_{im,j} = \hat{w}^l_{im,j} - w_-$. Similarly, the biases of the neurons $\boldsymbol{\alpha}$ represent linearly translated values of the parameters $\hat{b}^l_j$, i.e. $b^l_j = \hat{b}^l_j - b_-$. The relation is defined through the following equations

$$\begin{aligned} w^l_{im,j} &= \hat{w}^l_{im,j} - \hat{w}^l_{i0,j} - \hat{w}^0_{im,1} + \hat{w}^0_{i0,1}, \text{ and} \\ b^l_j &= \hat{b}^l_j - \hat{b}^0_1 + \sum_{i=1}^{I} \left(\hat{w}^l_{i0,j} - \hat{w}^0_{i0,1} \right) \quad . \end{aligned} \tag{10}$$

This entails that the modification of synaptic weights and biases via synaptic and intrinsic plasticity during learning results in adaptation of the parameters of the represented generative model. Not all parameters in the generative model are learned, however. We assume that the parameters

$$\begin{aligned} &\hat{w}^l_{i0,j} \text{ for all } i, \text{ all } j \text{ and all } l \neq 0, \\ &\hat{w}^0_{im,1} \text{ for all } i, \text{ all } m \neq 0, \\ &\hat{w}^0_{i0,1} \text{ for all } i, \text{ and} \\ &\hat{b}^0_1, \end{aligned} \tag{11}$$

that do not have a corresponding synaptic weight or a bias, have fixed values that are not subject to learning. In particular, $\hat{w}^l_{i0,j}$, $\hat{w}^0_{i0,1}$ and $\hat{b}^0_1$ are assumed to have the following values

$$\begin{aligned} \hat{w}^l_{i0,j} &= -V \qquad \text{for all } l \neq 0, i \text{ and } j, \\ \hat{w}^0_{i0,1} &= -V \qquad \text{for all } i, \text{ and} \\ \hat{b}^0_1 &= -V \qquad , \end{aligned} \tag{12}$$

where V is a large positive constant. We assume that the constant V is large enough so that the probabilities $p(z, \mathbf{x}; \boldsymbol{\theta})$ where at least one of the variables z and x^i (for $i = 1, ..., I$) has zero value are

Table 1. Mathematical symbols used in the definition of the learning module.

Symbols related to the RVs of the inputs and the outputs	
$\mathbf{x}$	vector of all multinomial RVs $(x^1, \ldots, x^I)$ corresponding to the inputs
x^i	i-th multinomial RV from the vector $\mathbf{x}$
z	RV corresponding to the output neurons
$M(\ldots)$	operator that gives the maximum integer value of the RV given as an argument; For example $M(x^i)$ and $M(z)$ denote the maximum values of x^i and z respectively.
$p^*(\mathbf{x}, z)$	target probability distribution learned by the learning module
The output and input neurons in the learning module	
$\boldsymbol{\chi}^i$	population of input neurons that together encode the value of the RV x^i through population coding
χ^{im}	input neuron in $\boldsymbol{\chi}^i$ whose firing signals the value m of the RV x^i
x^{im}	binary RV that assumes value 1 if and only if $x^i = m$; It corresponds to the coding property of the input neuron χ^{im}.
$\boldsymbol{\zeta}$	population of output neurons that encode the value of the RV z
ζ^l	output neuron in $\boldsymbol{\zeta}$ whose firing signals the value l of the RV z
The WTA populations of neurons in the learning module and their associated RVs	
$\boldsymbol{\alpha}$	the whole WTA population of neurons that represent the auxiliary RVs $\boldsymbol{a}$

$\boldsymbol{\alpha}^l$	subpopulation of neurons in $\boldsymbol{\alpha}$ that connects to the output neuron ζ^l
J^l	number of neurons in $\boldsymbol{\alpha}^l$
α_j^l	A neuron from the subpopulation $\boldsymbol{\alpha}^l$
a_j^l	binary RV which value corresponds to the coding property of the neuron α_j^l
$\boldsymbol{a}^l$	vector of all RVs a_j^l (for all $j = 1, \ldots, J^l$) that corresponds to the subpopulation of neurons $\boldsymbol{\alpha}^l$
$\boldsymbol{a}$	vector of the union of the RVs in the vectors $\boldsymbol{a}^l$ for all $l = 0, \ldots, L$; Corresponds to the WTA population $\boldsymbol{\alpha}$

Synaptic weights and biases and their corresponding parameters in the generative model

b_j^l	bias (intrinsic excitability) of the neuron α_j^l
$w_{im,j}^l$	synaptic weight of the synaptic connection that connects the input neuron χ^{im} to the neuron α_j^l
$p(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta})$	probability distribution of the generative model implicitly represented in the module
$\hat{b}_j^l$	parameter in the generative model $p(\mathbf{x}, z; \boldsymbol{\theta})$; Every such parameter, except for $l = 0$, is represented in the learning module by the bias b_j^l through the relation $b_j^l = \hat{b}_j^l - b_-$.
$\hat{w}_{im,j}^l$	parameter in the mixture generative model $p(\mathbf{x}, z; \boldsymbol{\theta})$; Every such parameter, except the ones with $l = 0$ or $m = 0$, is represented in the network by the synaptic weight $w_{im,j}^l$ through the relation $w_{im,j}^l = \hat{w}_{im,j}^l - w_-$.
$\boldsymbol{\theta}$	vector of all parameters of the generative model of the module; It includes all $\hat{b}_j^l$ (for all l and j) and all $\hat{w}_{im,j}^l$ (for all l, i, m and j) as components.

Indices used throughout all symbols

l	index that iterates through the output neurons ζ^l, and through their corresponding WTA subpopulations $\boldsymbol{\alpha}^l$ as well as through the binary RVs z^l

j	index that enumerates the individual neurons in the subpopulation $\boldsymbol{\alpha}^l$
i	index that iterates through the RVs x^i, and also through their corresponding populations of input neurons $\boldsymbol{\chi}^i$
m	index that enumerates the binary RVs x^{im} that represent individual values of the input RV x^i, and their corresponding input neurons χ^{im} in the population $\boldsymbol{\chi}^i$

Table 1. Contd. : Mathematical symbols used in the definition of the learning module.

much smaller than the probabilities of value assignments where all variables have non-zero values (see (5)).

The fixed parameters are all subsumed in the baseline values w_- and b_-:

$$\begin{aligned} w_- &= \hat{w}^l_{i0,j} + \hat{w}^0_{im,1} - \hat{w}^0_{i0,1} \qquad \text{for all } i,\, j,\, l \neq 0 \text{ and } m \neq 0, \text{ and} \\ b_- &= \hat{b}^0_1 - \sum_{i=1}^{I} \left(\hat{w}^l_{i0,j} - \hat{w}^0_{i0,1}\right) \qquad \text{for all } l \text{ and } j \quad . \end{aligned} \tag{13}$$

From this it follows that one can write the synaptic weights and the biases as

$$\begin{aligned} w^l_{im,j} &= \hat{w}^l_{im,j} - w_-, \\ b^l_j &= \hat{b}^l_j - b_- \end{aligned} \tag{14}$$

Hence, we have a one-to-one mapping between the synaptic weights and biases of the neurons $\boldsymbol{\alpha}$, and the parameters in the generative model that are subject to adaptation during learning. Note that from (12) and (13) it follows that

$$\begin{aligned} \hat{w}^0_{im,1} &= w_- \qquad \text{for all } i \text{ and } m \neq 0, \text{ and} \\ \hat{b}^0_1 &= b_- \qquad . \end{aligned} \tag{15}$$

If we now substitute the synaptic weights and biases from (10) in the expression for the firing probability density $\rho^l(t)$ of the output neuron ζ^l from (4) we obtain

$$\rho^l(t) = \frac{1}{\tau} \cdot \frac{\sum_{j=1}^{J^l} \exp\left(\hat{b}^l_j + \sum_{i=1}^{I} \hat{w}^l_{i0,j} + \sum_{i=1}^{I} \sum_{m=1}^{M(x^i)} (\hat{w}^l_{im,j} - \hat{w}^l_{i0,j}) x^{im}(t)\right)}{\exp\left(\hat{b}^0_1 + \sum_{i=1}^{I} \hat{w}^0_{i0,1} + \sum_{i=1}^{I} \sum_{m=1}^{M(x^i)} (\hat{w}^0_{im,1} - \hat{w}^0_{i0,1}) x^{im}(t)\right)} \quad . \tag{16}$$

In this expression we can combine the two sums over the index i that iterates through $1, \ldots, I$ (we do this both in the denominator and in the numerator), and by using the equality $1 - \sum_{m=1}^{M(x^i)} x^{im}(t) = x^{i0}(t)$ (where x^{i0} is a binary RV such that $x^{i0} = 1$ if and only if $x^i = 0$), we get the simplified form

$$\rho^l(t) = \frac{1}{\tau} \cdot \frac{\sum_{j=1}^{J^l} \exp\left(\hat{b}^l_j + \sum_{i=1}^{I} \sum_{m=0}^{M(x^i)} \hat{w}^l_{im,j} x^{im}\right)}{\exp\left(\hat{b}^0_1 + \sum_{i=1}^{I} \sum_{m=0}^{M(x^i)} \hat{w}^0_{im,1} x^{im}\right)} \quad , \tag{17}$$

where we write simply x^{im} instead of $x^{im}(t)$, as we assume it is implicitly understood that it is a function of time. It is clear that the denominator is equal to $p(z=0,\mathbf{x};\boldsymbol{\theta})$. In the numerator we can rewrite the sum by iterating over values of the RV $\boldsymbol{a}$ as

$$\rho^l(t) = \frac{1}{\tau}\cdot\frac{\sum_{\boldsymbol{a}} p(z=l|\boldsymbol{a})\exp\left(\sum_{l',j}\hat{b}_j^{l'}a_j^{l'}+\sum_{l',j}\sum_{i=1}^{I}\sum_{m=0}^{M(x^i)}\hat{w}_{im,j}^{l'}x^{im}a_j^{l'}\right)}{p(z=0,\mathbf{x};\boldsymbol{\theta})\,A(\boldsymbol{\theta})} \tag{18}$$

where the sum indexed by $\boldsymbol{a}$ iterates over all possible values of the compound RV $\boldsymbol{a}$ having only one $a_j^l=1$ and others equal to 0, and in both sums inside the exp function indexed by l' and j, the index l' iterates over all possible values of z which are $\{0,\ldots,L\}$, and j iterates from 1 to $J^{l'}$. It is now easy to see that the numerator is equal to $p(z=l,\mathbf{x};\boldsymbol{\theta})\,A(\boldsymbol{\theta})$, and finally we obtain

$$\rho^l(t) = \frac{1}{\tau}\cdot\frac{p(z=l|\mathbf{x}(t);\boldsymbol{\theta})}{p(z=0|\mathbf{x}(t);\boldsymbol{\theta})} \quad . \tag{19}$$

Thus, the firing probability of the output neuron l is proportional to $p(z=l|\mathbf{x};\boldsymbol{\theta})$, and if the internal model $p(\mathbf{x},z;\boldsymbol{\theta})$ is close to $p^*(\mathbf{x},z)$, it will be proportional to $p^*(z=l|\mathbf{x})$ (as pointed out in section "A network module for learning stochastic associations" of Results). For how this result is used in the theory for networks of interconnected learning modules that learn to perform probabilistic inference in larger distributions see the section "Theoretical properties of networks of recursively interconnected basic learning modules" of Methods.

Deriving the probability distribution of the firing of the neurons $\boldsymbol{\alpha}$. We show in this subsection that the firing of the neurons $\boldsymbol{\alpha}$ in the WTA circuit during the presentation of an example $\langle\tilde{\mathbf{x}}(n),\tilde{z}(n)\rangle$ generates samples from the distribution $p(\boldsymbol{a}|\tilde{\mathbf{x}}(n),\tilde{z}(n);\boldsymbol{\theta})$. This result is used in section "The plasticity rules minimize the KL divergence through Expectation Maximization" of Methods where the link between the plasticity rules, the activity of the neurons, and different calculations of the EM algorithm are discussed. In particular, it is explained there that, as the RVs $\boldsymbol{a}$ are the hidden variables in the probabilistic model, the samples generated by the activity of the $\boldsymbol{\alpha}$ neurons together with the presented examples, form samples from the complete data distribution, which represents the expectation step of the EM algorithm. The result is further used in section "Proof that the plasticity rules minimize the objective function $\mathcal{U}(\boldsymbol{\theta})$ through Expectation Maximization" of Methods in that the proof is based on analyzing mean weight updates of the plasticity rules over the complete data distribution.

During the presentation of a single example, in the periods when it is not inhibited by the lateral inhibition due to a spike by another neuron in $\boldsymbol{\alpha}$, the firing probability density of each neuron α_j^l in $\boldsymbol{\alpha}$ ideally remains constant over time. Let us denote the stationary distribution of its associated RV a_j^l during the presentation of the example $\langle\tilde{\mathbf{x}}(n),\tilde{z}(n)\rangle$ with $p_{(n)}(a_j^l)$. We will also here use the assumption that the total firing rate of the $\boldsymbol{\alpha}^l$ neurons is very high upon presentation of an example with $\tilde{z}(n)=l$ irrespective of what the input $\tilde{\mathbf{x}}(n)$ is. That this assumption is true can be seen from the following. As we stated previously, we assume that the initial values of the biases of the $\boldsymbol{\alpha}$ neurons are high. This implies that in the beginning of learning, for each presented example there will be active $\boldsymbol{\alpha}$ neurons that fire in response to the example. In addition, as b_- has a large negative value (see (13) and (15)), it follows that the plasticity rule for the biases defined in (2) and (3) will drive and stabilize the biases of the active $\boldsymbol{\alpha}$ neurons towards very large values. Thus, when an example with $\tilde{z}(n)=l$ is presented, there will be always a subset of the neurons in $\boldsymbol{\alpha}$ that have high firing rates. Consequently, during the presentation of the example almost all the time there will be a neuron in $\boldsymbol{\alpha}^l$ active and $p_{(n)}(a_1^0=1)\approx 0$.

If we use the fact that $p_{(n)}(a_1^0=1)\approx 0$ for $p_{(n)}(a_j^l)$ we can write

$$p_{(n)}(a_j^l=1) = \frac{\rho_j^l}{\sum_{j'=1}^{J^l}\rho_{j'}^l} \quad \text{if} \quad \tilde{z}(n)=l \quad , \tag{20}$$

and equal to 0 otherwise. Here ρ_j^l denotes the firing probability density of α_j^l, which is constant during the presentation of the example. If we substitute now the firing probability density in (20) with

$$\rho_j^l = \frac{1}{\tau} \exp\left(b_j^l + \sum_{i=1}^{I} \sum_{m=1}^{M(x^i)} w_{im,j}^l \tilde{x}^{im}(n) \right) \quad , \tag{21}$$

and additionally substitute the synaptic weights and biases with their related model parameters according to (10), we get

$$p_{(n)}(a_j^l = 1) = \frac{\exp\left(\hat{b}_j^l - \hat{b}_1^0 + \sum_{i=1}^{I} \sum_{m=0}^{M(x^i)} (\hat{w}_{im,j}^l - \hat{w}_{im,1}^0) \tilde{x}^{im}(n) \right)}{\sum_{j'=1}^{J^l} \exp\left(\hat{b}_{j'}^l - \hat{b}_1^0 + \sum_{i=1}^{I} \sum_{m=0}^{M(x^i)} (\hat{w}_{im,j'}^l - \hat{w}_{im,1}^0) \tilde{x}^{im}(n) \right)} \quad , \tag{22}$$

if $\tilde{z}(n) = l$ and 0 otherwise. Then if we use the definition of the generative model in (5), we can write both the numerator and denominator as

$$p_{(n)}(a_j^l = 1) = \frac{p(\boldsymbol{a} = (l,j), \tilde{\mathbf{x}}(n), \tilde{z}(n); \boldsymbol{\theta}) / p(\boldsymbol{a} = (0,1), \tilde{\mathbf{x}}(n); \boldsymbol{\theta})}{\sum_{j'=1}^{J^l} p(\boldsymbol{a} = (l,j'), \tilde{\mathbf{x}}(n), \tilde{z}(n); \boldsymbol{\theta}) / p(\boldsymbol{a} = (0,1), \tilde{\mathbf{x}}(n); \boldsymbol{\theta})} \quad . \tag{23}$$

By multiplying now both the numerator and denominator by $p(\boldsymbol{a} = (0,1), \tilde{\mathbf{x}}(n), \tilde{z}(n); \boldsymbol{\theta})$ and by using the fact that

$$\sum_{j'=1}^{J^l} p(\boldsymbol{a} = (l,j'), \tilde{\mathbf{x}}(n), \tilde{z}(n); \boldsymbol{\theta}) = p(\tilde{\mathbf{x}}(n), \tilde{z}(n); \boldsymbol{\theta}) \tag{24}$$

we finally arrive at

$$p_{(n)}(a_j^l = 1) = p(\boldsymbol{a} = (l,j) | \tilde{\mathbf{x}}(n), \tilde{z}(n); \boldsymbol{\theta}) \quad . \tag{25}$$

Hence, the neurons $\boldsymbol{\alpha}$ sample from $p(\boldsymbol{a} | \tilde{\mathbf{x}}(n), \tilde{z}(n); \boldsymbol{\theta})$. As we pointed out at the beginning, this derivation is an important step in the proof that the plasticity rules implement the Expectation Maximization algorithm (see next subsection).

The plasticity rules minimize the KL divergence through Expectation Maximization. Here we further discuss the key property of the learning module pointed out in section "A network module for learning stochastic associations" of Results, that the plasticity rules modify the synaptic weights and biases in a way that minimizes the KL divergence between the full joint distribution of the generative model $p(z, \mathbf{x}; \boldsymbol{\theta})$ and the corresponding target joint distribution $p^*(z, \mathbf{x})$. We denote the KL divergence between $p(z, \mathbf{x}; \boldsymbol{\theta})$ and $p^*(z, \mathbf{x})$ as the objective function $\mathcal{U}(\boldsymbol{\theta})$ i.e.

$$\mathcal{U}(\boldsymbol{\theta}) = D_{KL}\left(p^*(z, \mathbf{x}) || p(z, \mathbf{x}; \boldsymbol{\theta}) \right) \quad , \tag{26}$$

By using (26) the key property of the learning module can be reformulated in form of the following theorem

Theorem 1. *The synaptic and intrinsic plasticity rules introduced in (1), (2) and (3) change the parameters $\boldsymbol{\theta}$ so that they always converge to a local minimum of $\mathcal{U}(\boldsymbol{\theta})$ subject to the normalization constraints (6).*

The complete proof of the theorem is given in the next section "Proof that the plasticity rules minimize the objective function $\mathcal{U}(\boldsymbol{\theta})$ through Expectation Maximization" of Methods. In this section we discuss the approach and the main steps of the proof, and we also describe how the convergence is carried out through the EM algorithm, i.e. how the firing of the neurons and the synaptic weight updates by the

plasticity rules implement different steps of EM. The complete details to the EM implementation can be found in section "Proof that the plasticity rules minimize the objective function $\mathcal{U}(\boldsymbol{\theta})$ through Expectation Maximization" of Methods.

The proof of Theorem 1 utilizes results from the Robbins-Monro methods for stochastic approximation (Kushner and Yin, 2003). What is shown in the proof is that first, STDP and intrinsic plasticity drive the parameters $\boldsymbol{\theta}$ towards satisfying the normalization constraints in (6). As a second step, it is shown that given that $\boldsymbol{\theta}$ satisfy the normalization constraints, then the mean update of the parameters averaged over many presented examples is in the direction of the negative gradient of the objective function $\mathcal{U}(\boldsymbol{\theta})$. From these two intermediate steps, it follows that the parameters converge to a local minimum of $\mathcal{U}(\boldsymbol{\theta})$ under the normalization constraints.

By reducing the KL difference between $p^*(z, \mathbf{x})$ and $p(z, \mathbf{x}; \boldsymbol{\theta})$, the module also aims to approximate the target conditional $p^*(z|\mathbf{x})$ by its internal model conditional $p(z|\mathbf{x}; \boldsymbol{\theta})$ that defines the firing of the output neurons. In particular, the objective function $\mathcal{U}(\boldsymbol{\theta})$ is an upper bound of the function

$$\mathcal{J}(\boldsymbol{\theta}) = \langle D_{KL}(p^*(z|\mathbf{x})||p(z|\mathbf{x}; \boldsymbol{\theta}))\rangle_{p^*(\mathbf{x})} \quad , \tag{27}$$

which is the mean KL divergence between the target conditional and the model conditional, averaged over the distribution $p^*(\mathbf{x})$ over the inputs. This is easy to see from the equality

$$\mathcal{U}(\boldsymbol{\theta}) = \mathcal{J}(\boldsymbol{\theta}) + D_{KL}(p^*(\mathbf{x})||p(\mathbf{x}; \boldsymbol{\theta})) \quad . \tag{28}$$

Thus, EM aims to reduce the measure of how much the two conditionals differ $\mathcal{J}(\boldsymbol{\theta})$ through minimizing its upper bound $\mathcal{U}(\boldsymbol{\theta})$. If the upper bound during learning decreases, this does not always mean $\mathcal{J}(\boldsymbol{\theta})$ will decrease as well. Nevertheless, if the upper bound $\mathcal{U}(\boldsymbol{\theta})$ converged during learning to a certain small value C, then after learning $\mathcal{J}(\boldsymbol{\theta}) \leq C$. And if C is small enough so that $p(z|\mathbf{x}; \boldsymbol{\theta})$ is a good approximation of $p^*(z|\mathbf{x})$, in such a way the minimization of the upper bound would lead to good learning of the conditional. Furthermore, in the simulation experiments we showed that the learning based on minimizing the upper bound $\mathcal{U}(\boldsymbol{\theta})$ works well.

The update of the synaptic weights and the biases of the neurons in the WTA circuit can be understood as performing an online stochastic approximation of the EM algorithm. The EM algorithm is an iterative optimization algorithm that finds a local minimum of $\mathcal{U}(\boldsymbol{\theta})$ indirectly through another function called the complete data log-likelihood. In order to establish the link with the neural network learning, we will consider a stochastic version of the EM algorithm (Jank, 2006; Wei and Tanner, 1990), that we describe in the following. We will also assume the more common version used in the literature where we have a finite sequence of examples $\langle \tilde{z}(n), \tilde{\mathbf{x}}(n)\rangle$ for $n = 1, \ldots, N$, instead of an infinite sequence drawn from $p^*(z, \mathbf{x})$. In the case of a finite sequence, the target distribution $p^*(z, \mathbf{x})$ is defined as the histogram of all examples. The main idea in EM is that we can substitute the original learning objective: to update the mixture generative model $p(z, \mathbf{x}; \boldsymbol{\theta})$ so that it gets close to $p^*(z, \mathbf{x})$ by another learning objective: to update the full (complete data) generative model $p(\boldsymbol{a}, \mathbf{x}, z; \boldsymbol{\theta})$ (together with the hidden RVs $\boldsymbol{a}$) to get closer to the complete data target distribution

$$p^*(\boldsymbol{a}, \mathbf{x}, z; \boldsymbol{\theta}) = p(\boldsymbol{a}|\mathbf{x}, z; \boldsymbol{\theta})p^*(\mathbf{x}, z) \quad . \tag{29}$$

As there are not any complete data samples to learn the generative model $p(\boldsymbol{a}, \mathbf{x}, z; \boldsymbol{\theta})$, but just incomplete data examples for the RVs $\mathbf{x}, z$, in stochastic EM one completes the data samples by generating sample values for the hidden RVs $\boldsymbol{a}$ from the generative model itself, i.e. through the conditional distribution $p(\boldsymbol{a}|\mathbf{x}, z; \boldsymbol{\theta})$. This is done for all examples $\tilde{\mathbf{x}}(n)$ for $n = 1, .., N$. Then one updates the parameters so that the likelihood that the complete data examples were generated by the complete data generative model $p(\boldsymbol{a}, \mathbf{x}, z; \boldsymbol{\theta})$ is increased. The completion of the data samples is called the "expectation" step, whereas the update of the parameters of the complete data generative model to increase the likelihood of the complete data examples is called the "maximization" step. The EM optimization is performed iteratively, where in each iteration one repeats these two steps.

We can link the above two steps in the EM algorithm to concrete mechanisms in the dynamics of the WTA circuit during learning. Specifically, the "expectation" step is implemented through the

firing of spikes by the neurons $\boldsymbol{\alpha}$ during the presentation of the data examples $\langle \tilde{\mathbf{x}}(n), \tilde{z}(n) \rangle$. Indeed, the firing of $\boldsymbol{\alpha}$ generates samples from the conditional distribution $p(\boldsymbol{a}|\mathbf{x}, z; \boldsymbol{\theta})$ (see subsection "Deriving the probability distribution of the firing of the neurons $\boldsymbol{\alpha}$" of section "Theoretical properties of the basic learning module (stochastic association module) and its plasticity" of Methods). The presented data examples together with the generated samples by the neurons $\boldsymbol{\alpha}$ form complete data samples from p^* in (29). The plasticity rules update the synaptic weights and biases exactly based on these complete data samples encoded in the spikes. Furthermore, given that the normalization constraints hold, the mean update of the parameters over the infinite sequence of examples (where the same finite sequence is repeated in succession infinitely) is in the direction of the gradient of the complete data log-likelihood (for details see next section "Proof that the plasticity rules minimize the objective function $\mathcal{U}(\boldsymbol{\theta})$ through Expectation Maximization" of Methods). In view of this established link between the offline mean update of the weights and the EM algorithm, we see that the online plasticity rules in the neural network implement an online stochastic approximation of the EM algorithm in the spirit of Monte Carlo EM (Jank, 2006; Wei and Tanner, 1990).

During learning of the distribution $p^*(\mathbf{x}|z = l)$ by the WTA circuit $\boldsymbol{\alpha}^l$, the neurons in $\boldsymbol{\alpha}^l$ self-organize so that each of them specializes to fire in response to one cluster of similar, frequently occurring input patterns. In the probability distribution $p^*(\mathbf{x}|z = l)$ this cluster corresponds to one mode of the distribution. A mode of the distribution can be described as a region of high probability in the probability space, surrounded by regions of low probability. The theoretical basis of the learning strategy via EM proves that the plasticity rules change the synaptic weights and biases to reduce the difference between the distribution of the inputs $p^*(\mathbf{x}|z = l)$ and the represented generative model $p(\mathbf{x}|z = l; \boldsymbol{\theta})$. At the end, the learning process yields a mixture distribution where each neuron in $\boldsymbol{\alpha}^l$ represents one mixture component in the form of the unimodal distribution $p(\mathbf{x}, a^l_j = 1; \boldsymbol{\theta})$ centered at one of the clusters of input patterns. The full generative model $p(\mathbf{x}|z = l; \boldsymbol{\theta})$ is then retrieved as a sum of the unimodal distributions. The unimodal mixture component is implicitly represented in the weights and the bias of the corresponding neuron, where the vector of synaptic weights of the neuron actually represents the center of the mixture component, i.e. the location in the input pattern space where its mode peaks with maximum probability.

Proof that the plasticity rules minimize the objective function $\mathcal{U}(\boldsymbol{\theta})$ through Expectation Maximization

We will give in this subsection a proof of Theorem 1 from the previous subsection "The plasticity rules minimize the KL divergence through Expectation Maximization" of Methods. The theorem captures the main property of the learning module discussed in section "A network module for learning stochastic associations" of Results, i.e. that the plasticity rules install in the module an internal representation of the stochastic associations between the variables $\mathbf{x}$ and z in the presented examples.

Before we present the main part of the proof, we first introduce some needed definitions and derivations.

Definitions and assumptions. According to the learning procedure, independent and identically distributed examples $\langle \tilde{\mathbf{x}}(0), \tilde{z}(0) \rangle$, $\langle \tilde{\mathbf{x}}(1), \tilde{z}(1) \rangle$, ..., $\langle \tilde{\mathbf{x}}(n), \tilde{z}(n) \rangle$, ... drawn from the target probability distribution are presented to the learning module one by one, each presented for a certain period of time Δt_E. The learning procedure was explained in section "Theoretical properties of the basic learning module (stochastic association module) and its plasticity" of Methods. The time interval Δt_E for the presentation of the examples should be several times larger than the duration of the PSPs τ. The reason for this is that at the beginning of a time period Δt_E when a new example $\langle \tilde{\mathbf{x}}(n+1), \tilde{z}(n+1) \rangle$ is presented to the module, the values of the RVs $\mathbf{x}$ determined by the firing of the input neurons $\boldsymbol{\chi}$ do not immediately change to the new example. Additionally, the neurons $\boldsymbol{\alpha}$ do not immediately start to sample from the new conditional distribution $p(\boldsymbol{a}|\tilde{\mathbf{x}}(n+1), \tilde{z}(n+1); \boldsymbol{\theta})$. There can be residual active EPSPs at the synaptic inputs connecting from the input neurons that encode the values $\tilde{\mathbf{x}}(n)$ from the previous time period Δt_E. Similarly, there can be residual EPSPs from active neurons in $\boldsymbol{\alpha}$ that prevent the other $\boldsymbol{\alpha}$ neurons to immediately start sampling correctly from the new conditional distributions $p(\boldsymbol{a}|\tilde{\mathbf{x}}(n+1), \tilde{z}(n+1); \boldsymbol{\theta})$.

Therefore, for a time period of duration of one EPSP τ at the beginning of the presentation of a new example, the update of the synaptic weights and biases might be incorrect. Nevertheless, if τ is several times smaller than Δt_E, this incorrect contribution to the learned weight (or bias) update should be insignificant.

As Δt_E is larger than τ, this means that for each example $\langle \tilde{\mathbf{x}}(n), \tilde{z}(n) \rangle$ the WTA circuit generates several samples from the conditionals $p(\boldsymbol{a}|\tilde{\mathbf{x}}(n), \tilde{z}(n); \boldsymbol{\theta})$ as the neurons $\boldsymbol{\alpha}$ spike several times during Δt_E (see subsection "Deriving the probability distribution of the firing of the neurons $\boldsymbol{\alpha}$" of section "Theoretical properties of the basic learning module (stochastic association module) and its plasticity" of Methods). For simplicity of notation and formulation, in the convergence proof we will assume that for each $\tilde{\mathbf{x}}(n)$ exactly one sample $\tilde{\boldsymbol{a}}(n)$ from $p(\boldsymbol{a}|\tilde{\mathbf{x}}(n), \tilde{z}(n); \boldsymbol{\theta})$ is generated. The convergence proof can be easily extended for the more general case when several samples are generated from $p(\boldsymbol{a}|\tilde{\mathbf{x}}(n), \tilde{z}(n); \boldsymbol{\theta})$. Given this assumption, the presented data example $\langle \tilde{\mathbf{x}}(n), \tilde{z}(n) \rangle$ together with the generated sample $\tilde{\boldsymbol{a}}(n)$ by the neurons $\boldsymbol{\alpha}$ represent one complete data example $\langle \tilde{\mathbf{x}}(n), z(n), \tilde{\boldsymbol{a}}(n) \rangle$ drawn from the complete data distribution $p^*(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta})$.

If with $\boldsymbol{\theta}(n)$ we denote the parameter values before applying the learning rule for the n-th example, and with $\boldsymbol{\theta}(n+1)$ we denote the values after the application of the learning rule, then

$$\boldsymbol{\theta}(n+1) = \Omega\left(\boldsymbol{\theta}(n) + \eta(n)\Delta\boldsymbol{\theta}(n)\right) \tag{30}$$

where $\eta(n)$ is the learning rate used in the n-th iteration, $\Delta\boldsymbol{\theta}(n)$ is the update according to the learning rules defined in (1), (2) and (3) (for easier reference, the synaptic and intrinsic plasticity rules are also given in Table 2), and $\Omega(\boldsymbol{\theta})$ is a function that clips the parameter values within the intervals $-w_{min} \leq \hat{w}^l_{im,j} \leq 0$ and $-b_{min} \leq \hat{b}^l_j \leq 0$. The set of values $\boldsymbol{\theta}$ that are within these intervals we will denote with $D(\boldsymbol{\theta})$. The subset of $D(\boldsymbol{\theta})$ where the normalization constraints (6) are satisfied we will denote with $C_{\boldsymbol{\theta}}$. As the update rule for the biases b^l_j in Table 2 is defined in continuous time, we need to transform it into a form consistent with the previously stated simplification that the updates are performed for one data example drawn from $p^*(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta})$, or in other words, that the population of neurons $\boldsymbol{\alpha}$ fires one spike during the presentation of a data example $\langle \tilde{\mathbf{x}}(n), \tilde{z}(n) \rangle$. We can do that easily by assuming that the examples are presented for a period of τ (corresponding to one spike sample), which yields the following update for the model parameters encoded in the biases

$$\Delta\hat{b}^l_j = \tau a^l_j \exp(-\hat{b}^l_j) - \tau \quad . \tag{31}$$

Although in Table 2 the learning rates are different for the weights and the biases, here for simplicity we assume that the learning rate is the same for all parameters. We additionally assume that the sequence of learning rates $\eta(n)$ satisfies

$$\begin{aligned} &\sum_{n=1}^{\infty} \eta(n) = \infty \text{ and} \\ &\sum_{n=1}^{\infty} \eta(n)^2 < \infty \quad . \end{aligned} \tag{33}$$

The initial values of the parameters $\boldsymbol{\theta}(0)$ are randomly drawn from $D(\boldsymbol{\theta})$.

As pointed out previously (see section "Theoretical properties of the basic learning module (stochastic association module) and its plasticity" of Methods), in the joint distribution $p^*(\mathbf{x}, z)$ of the examples zero values of z and x^i do not occur, i.e. $p^*(x^i = 0) = 0$ for all $i = 1, \ldots, I$, and additionally $p^*(z = 0) = 0$. In the implicitly represented generative model $p(\mathbf{x}, z; \boldsymbol{\theta})$ the probabilities where at least one of the RVs z and x^i $(i = 1, \ldots, I)$ has zero value are represented by the fixed parameters (see (11) and accompanying text) and are not learned. The fixed parameters assume values (12) so that the corresponding probabilities over zero values are very small, i.e. close to zero. Ideally, for the theoretical analysis we set these parameters so that

Synaptic plasticity At each postsynaptic spike of the neuron α_j^l, at time t, the synaptic weight $w_{im,j}^l$ undergoes an update $w_{im,j}^l \leftarrow w_{im,j}^l + \eta\, \Delta w_{im,j}^l$ where η is the learning rate and

$$\Delta w_{im,j}^l = \begin{cases} e^{-(w_{im,j}^l + w_-)} - 1, & \text{if } \chi^{im} \text{ fired in } [t-\tau, t]\ , \\ -1, & \text{if } \chi^{im} \text{ did not fire in } [t-\tau, t]. \end{cases}$$

The parameter w_- is a baseline parameter, and τ is a parameter that corresponds to the duration of PSPs.

Intrinsic plasticity

At the time of each spike of the neuron α_j^l the bias instantaneously changes its value according to $b_j^l \leftarrow b_j^l + \eta'\, \Delta b_j^l$, where

$$\Delta b_j^l = \tau e^{-(b_j^l + b_-)} \quad ,$$

η' is the learning rate and b_- is a baseline parameter for the bias. In addition, between spikes the bias exhibits continuous decay according to the differential equation

$$\dot{b}_j^l = -\eta' \quad . \tag{32}$$

Table 2. Synaptic and intrinsic plasticity rules in the neural network of the basic learning module.

$$\begin{aligned} &p(x^i = 0; \boldsymbol{\theta}) = 0 \text{ for all } i \in \{1, .., I\} \quad , \text{ and} \\ &p(z = 0; \boldsymbol{\theta}) = 0 \quad , \end{aligned} \tag{34}$$

in order to match the values $p^*(x^i = 0) = 0$ and $p^*(z = 0) = 0$. In simulations, the probabilities over zero values represent a very small, insignificant portion of the mass of the learned generative model. This is also true when learning with a network of interconnected learning modules (see section "Theoretical properties of networks of recursively interconnected basic learning modules" of Methods).

The assumption (34) can be easily achieved by letting in (12) the constant V to converge to $V \to +\infty$. Indeed, from $V \to +\infty$ it follows that $\hat{w}_{i0,j}^l \to -\infty$ and $\hat{b}_1^0 \to -\infty$. The weight offset w_- remains constant at any time during the limit process (see (12) and (13)). The offset value for the biases does not remain constant, however, it converges to $b_- \to -\infty$. To emulate this idealized condition in simulations it is sufficient to set b_- to a very large negative value. This ensures that the biases of the active $\boldsymbol{\alpha}$ neurons converge to large positive values, which then entails that there is always a non-empty subset of $\boldsymbol{\alpha}$ neurons that have high firing rates upon presentation of an example (see section "Deriving the probability distribution of the firing of the neurons $\boldsymbol{\alpha}$" of Methods for further details). High firing rates of the $\boldsymbol{\alpha}$ neurons imply $p(a = (0,1)|\mathbf{x}; \boldsymbol{\theta}) \approx 0$ and consequently $p(z = 0; \boldsymbol{\theta}) \approx 0$ in (34). In section "Theoretical properties of networks of recursively interconnected basic learning modules" of Methods we also demonstrate that after the convergence of the weights and biases during learning to a local optimum, large negative value of b_- implies $p(z = 0|\mathbf{x}; \boldsymbol{\theta}) \approx 0$, which is consistent with (34).

Finding a local minimum of $\mathcal{U}(\boldsymbol{\theta})$ through Expectation Maximization. We describe here the EM algorithm applied to the concrete case of finding a local minimum of the objective function $\mathcal{U}(\boldsymbol{\theta})$ in (26), without any reference to the learning module and how it is implemented there. After we give the convergence proof that the plasticity rules drive the synaptic weights and biases to a local minimum of

$\mathcal{U}(\boldsymbol{\theta})$, we return to the question what mechanisms in the dynamics of the learning module implement different steps of the EM algorithm.

The objective function $\mathcal{U}(\boldsymbol{\theta})$ can be written as

$$\mathcal{U}(\boldsymbol{\theta}) = -\mathcal{L}(\boldsymbol{\theta}) - \mathcal{S}_{p^*}(\mathbf{x}, z) \quad , \tag{35}$$

where

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{p^*(\mathbf{x},z)} \left[\log p(\mathbf{x}, z; \boldsymbol{\theta})\right] \tag{36}$$

is the log-likelihood function and

$$\mathcal{S}_{p^*}(\mathbf{x}, z) = -\sum_{\mathbf{x},z} p^*(\mathbf{x}, z) \log p^*(\mathbf{x}, z) \tag{37}$$

is the entropy of the RVs $\mathbf{x}$ and z with respect to the target joint distribution $p^*(\mathbf{x}, z)$. As $\mathcal{S}_{p^*}(\mathbf{x}, z)$ does not depend on the parameters $\boldsymbol{\theta}$, the constrained local minima of $\mathcal{U}(\boldsymbol{\theta})$ match the constrained local maxima of $\mathcal{L}(\boldsymbol{\theta})$.

In the following we will explain the steps of the EM algorithm that finds a local maximum of $\mathcal{L}(\boldsymbol{\theta})$. EM optimizes $\mathcal{L}(\boldsymbol{\theta})$ indirectly through another function called the complete data log-likelihood

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}) = \mathbb{E}_{p^*(\boldsymbol{a},\mathbf{x},z;\boldsymbol{\theta})} \left[\log p(\boldsymbol{a}, \mathbf{x}, z; \boldsymbol{\theta})\right] \tag{38}$$

where $p^*(\boldsymbol{a}, \mathbf{x}, z; \boldsymbol{\theta})$ is the complete data probability distribution in (29). It is an algorithm that updates the parameters $\boldsymbol{\theta}$ in iterations. We will denote the values of the parameters in current iteration as $\boldsymbol{\theta}_{old}$. Let $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}_{old})$ be the complete data log-likelihood $\tilde{\mathcal{L}}(\boldsymbol{\theta})$ where in the probability distribution in the expectation we have the parameter values $\boldsymbol{\theta}_{old}$ from the current iteration, i.e.

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}_{old}) = \mathbb{E}_{p^*(\boldsymbol{a},\mathbf{x},z;\boldsymbol{\theta}_{old})} \left[\log p(\boldsymbol{a}, \mathbf{x}, z; \boldsymbol{\theta})\right] \quad . \tag{39}$$

The iteration in the EM algorithm consists of two steps:

Expectation step: Calculate $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}_{old})$ as the expectation in (39).

Maximization step: Find parameter values $\boldsymbol{\theta}_{new}$ that satisfy

$$\mathcal{Q}(\boldsymbol{\theta}_{new}, \boldsymbol{\theta}_{old}) > \mathcal{Q}(\boldsymbol{\theta}_{old}, \boldsymbol{\theta}_{old}) \quad , \tag{40}$$

and then set $\theta_{old} \leftarrow \theta_{new}$.

This is a formulation of EM called the generalized EM algorithm (Dempster et al., 1977). The essence of the algorithm lies in the fact that (40) implies also that

$$\mathcal{L}(\boldsymbol{\theta}_{new}) > \mathcal{L}(\boldsymbol{\theta}_{old}) \quad , \tag{41}$$

which means that in each iteration we increase the value of the log-likelihood. In fact, it also holds that

$$\left.\frac{\partial \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}_{old})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{old}} = \mathbb{E}_{p^*(\boldsymbol{a},\mathbf{x},z;\boldsymbol{\theta}_{old})} \left[\frac{\partial}{\partial \boldsymbol{\theta}} \log p(\boldsymbol{a}, \mathbf{x}, z; \boldsymbol{\theta})\right] = \left.\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{old}} \quad . \tag{42}$$

We will refer to this equation when we show that the learning module implements an online stochastic approximation of the generalized EM algorithm outlined here (see text after the convergence proof).

Definition of an auxiliary function $\mathcal{K}(\boldsymbol{\theta})$ used in the convergence proof. We define here an additional function $\mathcal{K}(\boldsymbol{\theta})$ which will be used in the convergence proof. A useful property of $\mathcal{K}(\boldsymbol{\theta})$ is that within the parameter domain where the normalization constraints (6) are satisfied, it is equal to the log-likelihood function $\mathcal{L}(\boldsymbol{\theta})$. Hence, in order to demonstrate in the proof that the plasticity rules minimize $\mathcal{U}(\boldsymbol{\theta})$, it will be sufficient to show that the rules drive the weights and biases to satisfy the normalization constraints, and when the normalization constraints are satisfied, they maximize $\mathcal{K}(\boldsymbol{\theta})$.

The $\mathcal{K}(\boldsymbol{\theta})$ function is defined as follows

$$\mathcal{K}(\boldsymbol{\theta}) = \mathbb{E}_{p^*(\mathbf{x},z)}\left[\log q(\mathbf{x}, z; \boldsymbol{\theta})\right] \tag{43}$$

where $q(\mathbf{x}, z; \boldsymbol{\theta})$ is the marginal of the probability distribution

$$q(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta}) = p(z|\boldsymbol{a}) \frac{\exp\left(\sum_{i,m}\sum_{l,j} \hat{w}^l_{im,j}\, x^{im}\, a^l_j + \sum_{l,j} \hat{b}^l_j a^l_j\right)}{\hat{A}_0(\boldsymbol{\theta}) \prod_{l,j}\left[\hat{A}^l_{ij}(\boldsymbol{\theta})\right]^{a^l_j}} \quad . \tag{44}$$

The numerator of this distribution is the same as in the definition of the generative model $p(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta})$ in (5). The difference in the definition between p and q is the denominator. In the denominator in (44) $\hat{A}_0(\boldsymbol{\theta})$ is defined as

$$\hat{A}_0(\boldsymbol{\theta}) = \sum_l \sum_j \exp(\hat{b}^l_j) \quad , \tag{45}$$

whereas

$$\hat{A}^l_{ij}(\boldsymbol{\theta}) = \sum_m \exp(\hat{w}^l_{im,j}) \quad \text{for all } i \in \{1, \ldots, I\},\, j \in \{1, \ldots, J^l\} \text{ and } l \in \{0, \ldots, L\} \tag{46}$$

It can be easily seen that if the normalization constraints (6) are satisfied, i.e. if $\boldsymbol{\theta} \in C_{\boldsymbol{\theta}}$, then

$$q(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta}) = p(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta}) \quad , \tag{47}$$

from which it follows that

$$\mathcal{K}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) \qquad \text{for all } \boldsymbol{\theta} \in C_{\boldsymbol{\theta}} \quad . \tag{48}$$

The derivatives of $\mathcal{K}(\boldsymbol{\theta})$ with respect to the parameters $\hat{w}^l_{im,j}$ are

$$\frac{\partial \mathcal{K}(\boldsymbol{\theta})}{\partial \hat{w}^l_{im,j}} = \mathbb{E}_{q^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}\left[\frac{\partial}{\partial \hat{w}^l_{im,j}} \log q(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta})\right] \tag{49}$$

where $q^*(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta})$ is

$$q^*(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta}) = p^*(\mathbf{x}, z) q(\boldsymbol{a}|\mathbf{x}, z; \boldsymbol{\theta}) \quad . \tag{50}$$

If we now substitute $q(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta})$ in the term inside the expectation in the derivative (49) with (44) and simplify, we obtain

$$\frac{\partial \mathcal{K}(\boldsymbol{\theta})}{\partial \hat{w}^l_{im,j}} = \mathbb{E}_{q^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}\left[x^{im} a^l_j\right] - \mathbb{E}_{q^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}\left[a^l_j\right] \exp(\hat{w}^l_{im,j}) \frac{1}{\hat{A}^l_{ij}(\boldsymbol{\theta})} \quad . \tag{51}$$

Similarly, for the derivative of $\mathcal{K}(\boldsymbol{\theta})$ with respect to $\hat{b}^l_j$ we have the following

$$\frac{\partial \mathcal{K}(\boldsymbol{\theta})}{\partial \hat{b}^l_j} = \mathbb{E}_{q^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}\left[\frac{\partial}{\partial \hat{b}^l_j} \log q(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta})\right] \quad , \tag{52}$$

and if we substitute (44) we arrive at

$$\frac{\partial \mathcal{K}(\boldsymbol{\theta})}{\partial \hat{b}_j^l} = \mathbb{E}_{q^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}\left[a_j^l\right] - \exp(\hat{b}_j^l)\frac{1}{\hat{A}_0(\boldsymbol{\theta})} \quad . \tag{53}$$

Deriving the local minima of $\mathcal{U}(\boldsymbol{\theta})$ under the constraints. As we stated previously, the constrained local minima of $\mathcal{U}(\boldsymbol{\theta})$ match the constrained local maxima of the log-likelihood $\mathcal{L}(\boldsymbol{\theta})$. Additionally, the constrained local maxima $\mathcal{L}(\boldsymbol{\theta})$ also match the constrained local maxima of $\mathcal{K}(\boldsymbol{\theta})$, since $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{K}(\boldsymbol{\theta})$ for all $\boldsymbol{\theta} \in C_{\boldsymbol{\theta}}$. Here we derive the local maxima of $\mathcal{K}(\boldsymbol{\theta})$ under the normalization constraints in (6) in analytical form. The result is used in the main proof in subsection "The convergence theorem and the proof" of this section, where we will show that the sequence $\boldsymbol{\theta}(n)$ during learning converges exactly to these local maxima.

The local maxima of $\mathcal{K}(\boldsymbol{\theta})$ can be found by calculating the derivatives of the Lagrangian function of $\mathcal{K}(\boldsymbol{\theta})$

$$\Lambda(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathcal{K}(\boldsymbol{\theta}) + \lambda_0 \left(\sum_{l=1}^{L} \sum_{j} \exp(\hat{b}_j^l) - 1 \right) + \sum_{l=1}^{L} \sum_{j,i} \lambda_{ij}^l \left(\sum_{m=1}^{M(x^i)} \exp(\hat{w}_{im,j}^l) - 1 \right) \quad . \tag{54}$$

The set of constrained local maxima is a subset of the solutions of the following equations obtained by setting the derivatives of the Lagrangian to 0:

$$\begin{aligned} \frac{\partial \Lambda}{\partial \hat{b}_j^l} &= \mathbb{E}_{q^*(\mathbf{x},z,\mathbf{a};\boldsymbol{\theta})}\left[a_j^l\right] + (\lambda_0 - 1)\exp(\hat{b}_j^l) = 0 \qquad \text{for all } l \neq 0 \text{ and } j, \text{ and} \\ \frac{\partial \Lambda}{\partial \hat{w}_{im,j}^l} &= \mathbb{E}_{q^*(\mathbf{x},z,\mathbf{a};\boldsymbol{\theta})}\left[x^{im} a_j^l\right] + \\ &\quad + \left(\lambda_{ij}^l - \mathbb{E}_{q^*(\mathbf{x},z,\mathbf{a};\boldsymbol{\theta})}\left[a_j^l\right]\right)\exp(\hat{w}_{im,j}^l) = 0 \quad \text{for all } l \neq 0, j \text{ and } m \neq 0 \quad . \end{aligned} \tag{55}$$

By summing the left and right sides of all equations in which λ_0 occurs, and by using the equalities $\sum_{l\neq 0,j} a_j^l = 1$ and $\sum_{l\neq 0,j} \exp(\hat{b}_j^l) = 1$, we obtain a solution for the Lagrange multiplier equal to

$$\lambda_0 = 0 \quad . \tag{56}$$

Similarly, if we sum the left and right sides of all equations where λ_{ij}^l occurs, and if we use the equalities $\sum_{m\neq 0} x^{im} = 1$ and $\sum_{m\neq 0} \exp(w_{im,j}^l) = 1$, we arrive at

$$\lambda_{ij}^l = 0 \qquad \text{for all } l \neq 0, i \text{ and } j \quad . \tag{57}$$

Interestingly, as all Lagrangian multipliers are 0, the critical points of the Lagrangian are also critical points of $\mathcal{K}(\boldsymbol{\theta})$, which means that the constrained local maxima of $\mathcal{K}(\boldsymbol{\theta})$ are also its unconstrained local maxima.

Based on these solutions for the Lagrange multipliers, we get from (55) the following implicit solutions for the parameters $\hat{w}_{im,j}^l$ and $\hat{b}_j^l$

$$\begin{aligned} \hat{b}_j^l &= \log \mathbb{E}_{q^*(\mathbf{x},z,\mathbf{a};\boldsymbol{\theta})}[a_j^l] = \log q^*(a_j^l = 1; \boldsymbol{\theta}) \quad \text{and} \\ \hat{w}_{im,j}^l &= \log \frac{\mathbb{E}_{q^*(\mathbf{x},z,\mathbf{a};\boldsymbol{\theta})}[x^{im} a_j^l]}{\mathbb{E}_{q^*(\mathbf{x},z,\mathbf{a};\boldsymbol{\theta})}[a_j^l]} = \log q^*(x^{im} = 1 | a_j^l = 1; \boldsymbol{\theta}) \quad . \end{aligned} \tag{58}$$

where $q^*(a_j^l; \boldsymbol{\theta})$ and $q^*(x^{im}|a_j^l; \boldsymbol{\theta})$ are marginal and conditional distributions derived from $q^*(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta})$ in (50). Note that these are implicit solutions, since $q^*(a_j^l; \boldsymbol{\theta})$ and $q^*(x^{im}|a_j^l; \boldsymbol{\theta})$ depend on the parameters $\boldsymbol{\theta}$. It can be easily verified that the solutions of (58) fulfill the normalization constraints. Indeed

$$\sum_{l,j} \exp(\hat{b}_j^l) = \sum_{l \neq 0, j} q^*(a_j^l = 1; \boldsymbol{\theta}) = 1 \tag{59}$$

and also

$$\sum_{m} \exp(\hat{w}_{im,j}^l) = \sum_{m \neq 0} q^*(x^{im} = 1 | a_j^l = 1; \boldsymbol{\theta}) = 1 \quad . \tag{60}$$

Therefore, we can substitute in the implicit solutions the probability distribution $q^*(\mathbf{x}, z, \mathbf{a}; \boldsymbol{\theta})$ with $p^*(\mathbf{x}, z, \mathbf{a}; \boldsymbol{\theta})$ from (29), which yields

$$\begin{aligned} \hat{b}_j^l &= \log p^*(a_j^l = 1; \boldsymbol{\theta}) \qquad \text{and} \\ \hat{w}_{im,j}^l &= \log p^*(x^{im} = 1 | a_j^l = 1; \boldsymbol{\theta}) \quad . \end{aligned} \tag{61}$$

We will denote the set of all finite points $\boldsymbol{\theta}$ that satisfy (61) as G. It is assumed that the bounds of the parameters w_{min} and b_{min} are chosen so that $G \in D(\boldsymbol{\theta})$. The critical points of the Lagrangian in G are either local maxima, local minima or saddle points of the objective function $\mathcal{U}(\boldsymbol{\theta})$ in (26) under the normalization constraints in (6).

The convergence theorem and the proof. After introducing the necessary definitions and notations in the previous subsections, we use them here to restate the Theorem 1 from the subsection "The plasticity rules minimize the KL divergence through Expectation Maximization" of Methods in more technical terms.

Theorem 1*. *The infinite sequence $\boldsymbol{\theta}(n)$ in (30) converges with probability 1 to the set of local minima of $\mathcal{U}(\boldsymbol{\theta})$ in (26) subject to the normalization constraints (6) .*

Proof: We define a "mean limit" ordinary differential equation (ODE) that corresponds to the sequence $\boldsymbol{\theta}(n)$ as $\dot{\boldsymbol{\theta}} = \overline{h}(\boldsymbol{\theta})$, where $\overline{h}(\boldsymbol{\theta}) = \mathbb{E}_{p^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}[\Delta\boldsymbol{\theta}]$. Here $\Delta\boldsymbol{\theta}$ is defined with the plasticity rules in Table 2. As for all n the parameters $\boldsymbol{\theta}(n)$ remain in the bounded set $D(\boldsymbol{\theta})$, it follows that $\sup_n \mathbb{E}_{p^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}[\|\Delta\boldsymbol{\theta}\|] < \infty$ and $\sup_n \mathbb{E}_{p^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}[(\Delta\boldsymbol{\theta})^2] < \infty$.

In order to prove convergence, we use the stochastic convergence theorem 3.1 in Chapter 5 from (Kushner and Yin, 2003) which states that, under the above assumptions, the sequence $\boldsymbol{\theta}(0), \boldsymbol{\theta}(1), \ldots, \boldsymbol{\theta}(n), \ldots$ from (30) converges with probability 1 to the stable points of the limit set of the "mean limit" ODE for all initial conditions $\boldsymbol{\theta}(0) \in D(\boldsymbol{\theta})$. The limit set of the "mean limit" ODE with respect to a set of the initial conditions $\boldsymbol{\theta}(0) \in A$, which we denote as $F(A)$, is defined as

$$F(A) = \lim_{s \to \infty} \bigcup_{\boldsymbol{\theta}' \in A} \{\boldsymbol{\theta}(s'), s' \geq s : \boldsymbol{\theta}(0) = \boldsymbol{\theta}'\} \quad . \tag{62}$$

Convergence to the limit set $F(A)$ with probability 1 means that for all stochastic realizations of the sequence $\boldsymbol{\theta}(n)$ it holds that

$$\lim_{n \to \infty} \min_{\boldsymbol{\theta} \in F(A)} |\boldsymbol{\theta}(n) - \boldsymbol{\theta}| = 0 \quad . \tag{63}$$

The main part of the proof will be to show that $F(D(\boldsymbol{\theta})) = G$, i.e. that the limit set of the ODE with the set of initial values $D(\boldsymbol{\theta})$ is equal to the critical points of the Lagrangian $\Lambda(\boldsymbol{\theta}, \boldsymbol{\lambda})$ which are also critical points of $\mathcal{K}(\boldsymbol{\theta})$. We will show that in two steps. First we will show that all trajectories of the ODE asymptotically converge to the set $C_{\boldsymbol{\theta}}$ where the normalization constraints (6) are satisfied. Then in the second step we will show that all trajectories with initial conditions $\boldsymbol{\theta}(0) \in C_{\boldsymbol{\theta}}$ converge to G.

Let $c_0(\boldsymbol{\theta})$ and $c_{ij}^l(\boldsymbol{\theta})$ be the functions that express how much the sums in the normalization constraints (6) deviate from 1, i.e.

$$\begin{aligned} c_0(\boldsymbol{\theta}) &= \sum_l \sum_j \exp(\hat{b}_j^l) - 1 \\ c_{ij}^l(\boldsymbol{\theta}) &= \sum_m \exp(\hat{w}_{im,j}^l) - 1 \quad . \end{aligned} \tag{64}$$

For all points $\boldsymbol{\theta}$ in the set $C_{\boldsymbol{\theta}}$ where the constraints are satisfied we have $c_0(\boldsymbol{\theta}) = 0$ and $c_{ij}^l(\boldsymbol{\theta}) = 0$ (for all $l \neq 0$ and all i and j). We now analyze how the values of $c_0(\boldsymbol{\theta})$ and $c_{ij}^l(\boldsymbol{\theta})$ change when $\boldsymbol{\theta}$ progresses along a solution trajectory of the mean limit ODE. For that purpose we calculate the dot products between $\overline{h}(\boldsymbol{\theta})$ and the gradients of $c_0(\boldsymbol{\theta})$ and $c_{ij}^l(\boldsymbol{\theta})$. The non-zero components of the gradient of $c_0(\boldsymbol{\theta})$ are

$$\frac{\partial c_0}{\partial \hat{b}_j^l} = \exp(\hat{b}_j^l). \tag{65}$$

Similarly the gradient of $c_{ij}^l(\boldsymbol{\theta})$ has the following non-zero components

$$\frac{\partial c_{ij}^l}{\partial \hat{w}_{im,j}^l} = \exp(\hat{w}_{im,j}^l) \quad . \tag{66}$$

Thus, for the dot products we have

$$\overline{h}(\boldsymbol{\theta}) \cdot \frac{dc_0}{d\boldsymbol{\theta}} = \sum_{l \neq 0} \sum_j \left(\mathbb{E}_{p^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}[a_j^l] \exp(-\hat{b}_j^l) - 1 \right) \exp(\hat{b}_j^l) = -c_0(\boldsymbol{\theta}) \quad , \tag{67}$$

and

$$\begin{aligned} \overline{h}(\boldsymbol{\theta}) \cdot \frac{dc_{ij}^l}{d\boldsymbol{\theta}} = \sum_{m \neq 0} \Big(& \mathbb{E}_{p^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}[a_j^l x^{im}] \exp(-\hat{w}_{im,j}^l) - \\ & - \mathbb{E}_{p^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}[a_j^l] \Big) \exp(\hat{w}_{im,j}^l) = -p^*(a_j^l = 1; \boldsymbol{\theta})\, c_{ij}^l(\boldsymbol{\theta}) \quad , \end{aligned} \tag{68}$$

where $p^*(a_j^l = 1; \boldsymbol{\theta})$ is a marginal of $p^*(\mathbf{x}, z, \boldsymbol{a}; \boldsymbol{\theta})$. From these equations we can conclude that $\lim_{s\to\infty} c_0(\boldsymbol{\theta}(s)) = 0$ and $\lim_{s\to\infty} c_{ij}^l(\boldsymbol{\theta}(s)) = 0$ for all $l \neq 0$ and all i and j. Thus, because $\overline{h}(\boldsymbol{\theta})$ is continuous and differentiable in $D(\boldsymbol{\theta})$, it follows that the limit set of the ODE for initial conditions $\boldsymbol{\theta}(0) \in D(\boldsymbol{\theta}) \backslash C_{\boldsymbol{\theta}}$ is inside $C_{\boldsymbol{\theta}}$, i.e. $F(D(\boldsymbol{\theta}) \backslash C_{\boldsymbol{\theta}}) \subseteq C_{\boldsymbol{\theta}}$. Additionally, the differentiability and continuity of $\overline{h}(\boldsymbol{\theta})$ imply that $F(A) = F(F(A))$ and $F(A_1) \subseteq F(A_2)$ if $A_1 \subseteq A_2$, for all $A, A_1, A_2, \subseteq D(\boldsymbol{\theta})$. By using these properties, we can derive that

$$F(D(\boldsymbol{\theta}) \backslash C_{\boldsymbol{\theta}}) = F(F(D(\boldsymbol{\theta}) \backslash C_{\boldsymbol{\theta}})) \subseteq F(D(\boldsymbol{\theta}) \cap C_{\boldsymbol{\theta}}) \quad . \tag{69}$$

We continue now with the second step of the proof where we show that trajectories starting within $C_{\boldsymbol{\theta}}$ converge to the critical points of the Lagrangian $\Lambda(\boldsymbol{\theta}, \boldsymbol{\lambda})$. First, from equations (67) and (68) it follows that $\overline{h}(\boldsymbol{\theta}) \cdot \frac{dc_0}{d\boldsymbol{\theta}} = 0$ and $\overline{h}(\boldsymbol{\theta}) \cdot \frac{dc_{ij}^l}{d\boldsymbol{\theta}} = 0$ for all $\boldsymbol{\theta} \in C_{\boldsymbol{\theta}}$. This means that trajectories $\boldsymbol{\theta}(s)$ with $\boldsymbol{\theta}(0) \in C_{\boldsymbol{\theta}}$ remain in $C_{\boldsymbol{\theta}}$, i.e. $\boldsymbol{\theta}(s) \in C_{\boldsymbol{\theta}}$ for all $s \neq 0$. Furthermore, it is easy to show that the set of critical points G of the Lagrangian $\Lambda(\boldsymbol{\theta}, \boldsymbol{\lambda})$ written in implicit form in (61) is the same as the set of the stationary points of the "mean limit" ODE obtained as a solution of the equation $\dot{\boldsymbol{\theta}} = 0$. What remains to be shown is that the trajectory always converges with probability 1 to one of the stationary points in G, i.e. that for example it does not enter a limit cycle. We do that by analyzing how the function $\mathcal{K}(\boldsymbol{\theta})$ changes along the trajectory of the ODE by calculating the dot product between $\overline{h}(\boldsymbol{\theta})$ and the gradient $\frac{\partial \mathcal{K}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ given in (51) and (53). As for $\boldsymbol{\theta} \in C_{\boldsymbol{\theta}}$ it holds that $\hat{A}(\boldsymbol{\theta}) = 1$ and $\hat{A}_{ij}^l(\boldsymbol{\theta}) = 1$, the dot product is

$$\overline{h}(\boldsymbol{\theta}) \cdot \frac{\partial \mathcal{K}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{l \neq 0} \sum_{j} \exp(-\hat{b}_j^l) \left(\mathbb{E}_{p^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}[a_j^l] - \exp(\hat{b}_j^l) \right)^2 + \\ + \sum_{l \neq 0} \sum_{j} \sum_{m \neq 0} \exp(-\hat{w}_{im,j}^l) \left(\mathbb{E}_{p^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}[x^{im} a_j^l - \exp(\hat{w}_{im,j}^l) a_j^l] \right)^2 \quad \geq 0 \quad , \tag{70}$$

and becomes equal to zero for the set of stationary points $\dot{\boldsymbol{\theta}} = 0$ which is the same as the set G. Hence, the ODE trajectories do not have any limit cycles and always converge to the critical points of the Lagrangian G, i.e. $F(D(\boldsymbol{\theta}) \cap C_{\boldsymbol{\theta}}) = G$. Together with the conclusion (69) of the first step, we finally obtain that the limit set of the ODE is

$$F(D(\boldsymbol{\theta})) = F(D(\boldsymbol{\theta}) \backslash C_{\boldsymbol{\theta}}) \cup F(D(\boldsymbol{\theta}) \cap C_{\boldsymbol{\theta}}) = F(D(\boldsymbol{\theta}) \cap C_{\boldsymbol{\theta}}) = G \quad . \tag{71}$$

According to the stochastic convergence theorem 3.1 in Chapter 5 from (Kushner and Yin, 2003), the infinite sequence $\boldsymbol{\theta}(n)$ converges to the stable points of the limit set G. Stochastic fluctuations would drive the sequence to escape from the unstable points in the limit set. We have shown in (70) that $\mathcal{K}(\boldsymbol{\theta})$ increases along an ODE trajectory that starts within $\boldsymbol{\theta} \in C_{\boldsymbol{\theta}}$. This implies that also $\mathcal{L}(\boldsymbol{\theta})$ increases as $\mathcal{K}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta})$ for $\boldsymbol{\theta} \in C_{\boldsymbol{\theta}}$. Thus, it follows that the stable points of the ODE in G are the local constrained maxima of $\mathcal{L}(\boldsymbol{\theta})$, subject to the normalization constraints, which concludes the proof. ■

By using results from the proof, we can now more precisely relate the update of the synaptic weights and biases by the plasticity rules to the "maximization" step of the EM algorithm, as discussed in subsection "The plasticity rules minimize the KL divergence through Expectation Maximization" of Methods. In the WTA circuit the plasticity rules update the weights and biases online, after each spike of a neuron in $\boldsymbol{\alpha}$. If we instead consider an offline update where the updates are averaged over many spikes of the $\boldsymbol{\alpha}$ neurons (theoretically infinitely many), and then afterwards they are applied to the weights and biases, then this offline update of the parameter vector in the limit of infinite number of spikes would be equal to

$$\overline{h}(\boldsymbol{\theta}) = \mathbb{E}_{p^*(\mathbf{x},z,\boldsymbol{a};\boldsymbol{\theta})}[\Delta \boldsymbol{\theta}] \quad . \tag{72}$$

The vector $\overline{h}(\boldsymbol{\theta})$ is the gradient of the ODE trajectory, which means that for a sufficiently small learning rate the offline mean update will always increase the expected log-likelihood $\mathcal{L}(\boldsymbol{\theta})$ provided that $\boldsymbol{\theta} \in C_{\boldsymbol{\theta}}$. This follows from the fact that $\mathcal{L}(\boldsymbol{\theta})$ increases along the ODE trajectory as we have shown in the proof (see (70) and the text afterwards). Furthermore, the offline mean update increases $\mathcal{L}(\boldsymbol{\theta})$ by calculating an expectation over the complete data samples from the distribution (29) which represents the "expectation" step of the EM algorithm. In subsection "The plasticity rules minimize the KL divergence through Expectation Maximization" of Methods we already showed that the spikes of the neurons $\boldsymbol{\alpha}$ together with the presented examples represent the complete data samples drawn from (29). This clarifies that the offline mean update of the plasticity rules indeed implements the "maximization" step of the generalized EM algorithm as defined in (42) and (40). From this it follows that the online plasticity rules implement an online stochastic approximation version of the "maximization" step of EM.

The proof of Theorem 1 (reformulated in Theorem 1* in this subsection) completes the theory behind the learning mechanisms and learning capabilities of the learning module described in section "A network module for learning stochastic associations" of Results. In particular, it rigorously shows that the learning module learns, via the EM algorithm, an internal model of the probabilistic relations between a set of variables $\mathbf{x}$ and another variable z in the presented examples.

Theoretical properties of networks of recursively interconnected basic learning modules

This section contains additional details on the learning approach with networks of recursively connected learning modules, that we described in section "Recursive combinations of the basic learning module

enable efficient learning of complex distributions from examples" of Results. After presenting the learning procedure for such networks, we will formulate in this section Theorem 2 which contains the theoretical basis of learning with networks of learning modules. Theorem 2 is derived from Theorem 1 (see section "The plasticity rules minimize the KL divergence through Expectation Maximization" of Methods), and shows that the plasticity mechanisms in a network of modules minimize another objective function that pertains to the represented probabilistic model in the whole network. This minimization then leads to learning an internal model of the stochastic associations between all variables y^k of the presented examples.

The population of neurons that encodes the values of the RV $y^k(t)$ over time t we denote with $\boldsymbol{\nu}^k$ (these are the output neurons of the learning module $\mathcal{N}^k$). For each non-zero value l of y^k, there is a neuron ν^{kl} whose firing signals this value. In particular, a spike of the neuron ν^{kl} at time t sets the value of y^k to $y^k = l$ for a time period of duration τ, after which the value changes to $y^k = 0$. To ensure a valid value of y^k, no other neuron in $\boldsymbol{\nu}^k$ should fire in the time interval $[t, t+\tau)$. This is ensured by the lateral inhibition between the $\boldsymbol{\alpha}^k$ neurons which drive the $\boldsymbol{\nu}^k$ neurons to fire.

The network learns from presented examples $\tilde{\mathbf{y}}(0)$, $\tilde{\mathbf{y}}(1)$, ..., $\tilde{\mathbf{y}}(n)$, ... drawn from the target probability distribution $p^*(\mathbf{y})$. In the examples the RV y^k assumes an integer value drawn from the set $\{1, 2, \ldots, M(y^k)\}$. An example is presented to the network in form of injected currents in the neurons. These injected currents are assumed to originate from external neurons. The neurons in $\boldsymbol{\nu}^k$ are driven by the injected currents such that their firing reflects correctly the values of the RVs y^k in the current example $\tilde{\mathbf{y}}(n)$. More precisely, for $\tilde{y}^k(n) = l$ the neuron ν^{kl} receives strong positive current and fires with a high firing rate, whereas the other neurons in the population $\boldsymbol{\nu}^k$ receive a strong negative current which prevents them from firing. The neurons $\boldsymbol{\alpha}$ also have currents injected during learning, that give information about the current example, exactly in the same way as explained for a single learning module (see section "Theoretical properties of the basic learning module (stochastic association module) and its plasticity" of Methods) where the presented example $\langle \tilde{\mathbf{x}}(n), \tilde{z} \rangle$ is here for the learning module $\mathcal{N}^k$ equal to $\langle \tilde{\mathbf{y}}^{B(k)}(n), \tilde{y}^k(n) \rangle$ (see Fig. 5).

The mathematical symbols that are used in the definition of the neural network are listed in Table 3.

As we stated in the section "Theoretical properties of the basic learning module (stochastic association module) and its plasticity" of Methods, only the non-zero values of z represent meaningful states of the external environment and are learned. The probability of the zero value $p(z = 0|\mathbf{x}; \boldsymbol{\theta})$ is not learned, and converges always to a very small value during learning. This can be easily seen if we express $p(z = 0|\mathbf{x}; \boldsymbol{\theta})$ through the firing probability densities $\rho_j^l(\mathbf{x})$ of the $\boldsymbol{\alpha}$ neurons given input $\mathbf{x}$

$$p(z = 0|\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{1 + \sum_{l=1}^{L} \sum_{j=1}^{J^l} \rho_j^l(\mathbf{x})} \quad . \tag{73}$$

From the inequality $\rho_j^l(\mathbf{x}) = \frac{1}{\tau} \exp(u_j^l(\mathbf{x})) \geq \frac{1}{\tau} \exp(b_j^l)$ it follows that

$$p(z = 0|\mathbf{x}; \boldsymbol{\theta}) \leq \frac{1}{1 + \sum_{l=1}^{L} \sum_{j=1}^{J^l} \exp(b_j^l)} \quad . \tag{74}$$

If we substitute (14) for the biases and use $b_- = -V$, which follows from (12) and (15), we obtain

$$p(z = 0|\mathbf{x}; \boldsymbol{\theta}) \leq \frac{\exp(-V)}{\sum_{l=0}^{L} \sum_{j=1}^{J^l} \exp(\hat{b}_j^l)} \quad . \tag{75}$$

According to the proof of Theorem 1* (see (67)), during learning the plasticity rules drive the synaptic weights and biases of the $\boldsymbol{\alpha}$ neurons towards satisfying the normalization constraints in (6) which simplifies (75) to $p(z = 0|\mathbf{x}; \boldsymbol{\theta}) \leq \exp(-V)$. As V is a very large positive constant (see (12) and accompanying text), we finally arrive at

$$p(z = 0|\mathbf{x}; \boldsymbol{\theta}) \approx 0 \quad . \tag{76}$$

Hence, we have shown that after learning, the sum of the firing probabilities of the output $\boldsymbol{\zeta}$ neurons is always very high (when inhibition is not active), regardless of the current input $\mathbf{x}$. Similarly, in case of a network of learning modules, after learning the output $\boldsymbol{\nu}^k$ neurons in each module in the network are in strong competition and $p^k(y^k = 0|\mathbf{y}^{B(k)}; \boldsymbol{\theta})$ is very small. From this it follows that in the stochastic dynamics of the spiking neural network states where there is a RV y^k with value $y^k = 0$ occur for very short time intervals, i.e. they have very small probability in the internal model distribution $p(\mathbf{y}; \boldsymbol{\theta})$. Note that in the examples drawn from the target probability distribution p^* such states are not present as the RVs y^k in the examples have values from 1 to $M(y^k)$. The zero values occur only in the neural network dynamics and they do not refer to meaningful values of variables in the external environment.

In order to show that if each learning module $\mathcal{N}^k$ learns an approximation of $p^*(y^k, \mathbf{y}^{B(k)})$ in its internal model $p^k(y^k, \mathbf{y}^{B(k)}; \boldsymbol{\theta}^k)$ then the stationary distribution of the whole network is close to $p^*(\mathbf{y})$, we use as a theoretical basis the neural computability condition (NCC) in (Pecevski et al., 2011; Buesing et al., 2011). The NCC has been identified there as a sufficient condition for representing a particular distribution p^* as stationary distribution of a network of spiking neurons. If the NCC is satisfied, then the stochastic dynamics of the resulting network produces (after some burn-in phase, whose length depends on the choice of its initial state) spontaneously network states (samples) that are drawn according to p^*. The network state is represented here as the vector of the RV values $\mathbf{y}(t)$ at time t, where the value of each RV $y^k(t)$ at time t is encoded in the spikes of the population of neurons $\boldsymbol{\nu}^k$ as described above. In order to have p^* as a stationary distribution of network states, the NCC requires that the probability density of ν^{kl} should be equal to

$$\rho^{kl}(t) = \frac{1}{\tau} \frac{p^*\left(y^k = l|\mathbf{y}^{B(k)}(t)\right)}{p^*\left(y^k = 0|\mathbf{y}^{B(k)}(t)\right)} \quad , \tag{77}$$

if the current value of y^k is $y^k(t) = 0$, and otherwise the neuron should be silent. If $p^*\left(y^k = 0|\mathbf{y}^{B(k)}(t)\right) \to 0$, as we assumed is the case in the presented examples, the NCC transforms as follows: the probability density of ν^{kl} at time t should be proportional to $p^*(y^k = l|\mathbf{y}^{B(k)}(t))$

$$\rho^{kl}(t) \sim p^*\left(y^k = l|\mathbf{y}^{B(k)}(t)\right) \quad , \tag{78}$$

if $y^k(t) = 0$, and should be 0 otherwise. Additionally, the firing probability densities ρ^{kl} for all principal neurons in the population $\boldsymbol{\nu}^k$ should be large enough so that if the spike of the last neuron was at time t, a new neuron fires almost immediately after $t + \tau$, i.e. the in-between time period where $y^k(t) = 0$ should be very short (ideally its duration should be equal to 0).

The firing probability density of the neuron ν^{kl} in the network is, according to (19), equal to

$$\rho^{kl}(t) = \frac{1}{\tau} \cdot \frac{p^k(y^k = l|\mathbf{y}^{B(k)}(t); \boldsymbol{\theta}^k)}{p^k(y^k = 0|\mathbf{y}^{B(k)}(t); \boldsymbol{\theta}^k)} \quad , \tag{79}$$

as it is one of the output neurons of the learning module $\mathcal{N}^k$. In addition, as we have shown above, the probability $p^k(y^k = 0|\mathbf{y}^{B(k)}(t); \boldsymbol{\theta}^k)$ after learning has a very small value, which makes the firing probabilities $\rho^{kl}(t)$ very high. We see that (79) has the same form as the firing probability (77) and (78) in the NCC, except that here we have the model conditional $p^k(y^k|\mathbf{y}^{B(k)}; \boldsymbol{\theta}^k)$ instead of the target conditional $p^*(y^k|\mathbf{y}^{B(k)})$ in the NCC. Thus, as the plasticity rules during learning change the weights and biases towards reducing the difference between $p^k(y^k, \mathbf{y}^{B(k)}; \boldsymbol{\theta}^k)$ and $p^*(y^k, \mathbf{y}^{B(k)})$, which is an upper bound of the difference between the corresponding conditionals (see (28)), this then should bring the firing probability $\rho^{kl}(t)$ from (79) closer to the desired firing probability in the NCC (78).

The objective function $\hat{\mathcal{U}}(\boldsymbol{\theta})$ for learning in the whole network can be formulated as the sum of the objective functions $\mathcal{U}(\boldsymbol{\theta})$ of all learning modules $\mathcal{N}^k$

$$\hat{\mathcal{U}}(\boldsymbol{\theta}) = \sum_k D_{KL}\left(p^*(y^k, \mathbf{y}^{B(k)})||p^k(y^k, \mathbf{y}^{B(k)}; \boldsymbol{\theta}^k)\right) \quad . \tag{80}$$

Using $\hat{\mathcal{U}}(\boldsymbol{\theta})$ we can state the following theorem for learning in the whole network:

Table 3. Mathematical symbols used in the definition of the network of interconnected learning modules.

Symbols related to the RVs in the target probability distribution	
$\mathbf{y}$	vector of all multinomial RVs $(y^1, \ldots, y^K)$ from the target probability distribution $p^*(\mathbf{y})$
y^k	k-th multinomial RV from the vector $\mathbf{y}$
$M(y^k)$	the maximum integer value the multinomial RV y^k can assume
$p^*(\mathbf{y})$	target probability distribution learned by the neural network
$\mathbf{y}^{B(k)}$	vector of the RVs from $\mathbf{y}$ that are in the Markov blanket of the RV y^k
$\mathcal{N}^k$	the learning module in the network that approximates the r.h.s of the NCC in (77) for the neurons $\boldsymbol{\nu}^k$ encoding the RV y^k
Output (and input) neurons of the modules $\mathcal{N}^k$ and their associated RVs	
$\boldsymbol{\nu}^k$	population of neurons in the network that together encode the value of the RV y^k through population coding
ν^{kl}	neuron in $\boldsymbol{\nu}^k$ whose firing signals the value l of the RV y^k
y^{kl}	binary RV that assumes value 1 if and only if $y^k = l$; It corresponds to the coding property of the neuron ν^{kl}.
Other symbols	
k	superscript index used to indicate that the symbol describes an element of the learning module $\mathcal{N}^k$
$\boldsymbol{\alpha}^k$, $\boldsymbol{\alpha}^{kl}$, α_j^{kl}, J^{kl}, a_j^{kl}, $\boldsymbol{a}^{kl}$, $\boldsymbol{a}^k$	symbols for the neurons in $\boldsymbol{\alpha}$ and their associated RV have the same meaning as the symbols for a single learning module in Table 1; The additional superscript index k indicates that the element belongs to the module $\mathcal{N}^k$.

b_j^{kl}, $w_{im,j}^{kl}$, $\hat{w}_{im,j}^{kl}$, $\hat{b}_j^{kl}$, $\boldsymbol{\theta}^k$	synaptic weights, biases and the parameters of the generative models have the same meaning as the symbols for a single learning module in Table 1; The additional superscript k identifies that the element belongs to the module $\mathcal{N}^k$.
$p^k(\mathbf{y}^{B(k)}, y^k; \boldsymbol{\theta}^k)$	probability distribution of the mixture generative model implicitly represented in the module $\mathcal{N}^k$
$\boldsymbol{\theta}$	vector of union of all parameters in the neural network $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}^k)$ from all generative models (corresponding to the learning modules $\mathcal{N}^k$)

Table 3. Contd.: Mathematical symbols used in the definition of the network of interconnected learning modules.

Theorem 2. *The synaptic and intrinsic plasticity rules in (1), (2) and (3) change the parameters $\boldsymbol{\theta}$ in a way so that they always converge to a local minimum of $\hat{\mathcal{U}}(\boldsymbol{\theta})$ subject to the normalization constraints (6).*

Theorem 2 directly follows from Theorem 1 and from the fact that each of the KL divergences in the sum are parametrized by a separate set of parameters $\boldsymbol{\theta}^k$. If $\hat{\mathcal{U}}(\boldsymbol{\theta}) = 0$ then according to the NCC we have that $p(\mathbf{y}; \boldsymbol{\theta}) = p^*(\mathbf{y})$. If the parameters converge to a good local minimum of $\hat{\mathcal{U}}(\boldsymbol{\theta})$ so that the learning modules in the network learn a good approximation of the firing probability densities of the NCC, then the stationary probability distribution $p(\mathbf{y}; \boldsymbol{\theta})$ of the network would approximate well the target distribution $p^*(\mathbf{y})$.

During its spontaneous activity the neural network enters states where there is at least one RV y^k with zero value. These states are not part of the target distribution p^*. Nevertheless, as that the output neurons after learning fire with very high firing rates when not inhibited (see (73)-(76)), the states with zero values occur only for very short time intervals and should not affect the correct Markov chain Monte Carlo sampling process of the stochastic network. Therefore, if the output neurons of the modules learn to fire approximately according to the NCC (78) and fire with very high firing rates, the distribution of the network states during spontaneous activity $p(\mathbf{y}; \boldsymbol{\theta})$ should be a good approximation to $p^*(\mathbf{y})$, as we show in our computer simulations.

If it satisfies the NCC, the neural network can not only generate samples from p^*, but it can also perform via sampling probabilistic inference based on p^* (described in section "Flexible retrieval of learnt statistical information through probabilistic inference" of Results). In a typical probabilistic inference task evidence is provided for a subset of the RVs $\mathbf{y}_e$, i.e. their values are known, and one wants to calculate the posterior distribution $p^*(\mathbf{y}_s|\mathbf{y}_e)$ for some of the unknown RVs $\mathbf{y}_s$ given the evidence. In the neural network the evidence is presented by clamping the neurons with injected input currents. For example, if the value of the RV y^k is $y^k = l$, then positive current from exogenous neurons is injected in the neuron ν^{kl} such that it fires with high firing rate, whereas all other neurons in the population $\boldsymbol{\nu}^k$ are kept silent through a negative injected current. A basic property of the network is that when evidence is injected in it, it changes its dynamics such that the neurons for the unknown RVs $\mathbf{y}_s$ generate samples exactly from the posterior $p^*(\mathbf{y}_s|\mathbf{y}_e)$. The posterior probabilities can then be estimated simply by counting the generated samples.

Details to computer simulations

Details to the computer simulation in example 2. All computer simulations were carried out with NEVESIM, an event-based neural network simulator developed in C++ with a Python interface (Pecevski et al., 2014). The simulator NEVESIM builds on techniques developed in the simulator PCSIM (Pecevski

et al., 2009). In all simulations we used the stochastic point neuron model from (Buesing et al., 2011; Pecevski et al., 2011). The parameter τ that defines the encoding of values of the RVs by the spikes was equal to $\tau = 15$ms. The neurons had absolute refractory period of duration τ. The lateral inhibition in the WTA circuits was implemented with a population of 5 inhibitory neurons, where all neurons in the population $\boldsymbol{\alpha}$ connect to these inhibitory neurons with a synaptic weight $w_{e2i} = 80$. The bias of the inhibitory neurons in the lateral inhibition was set to $b_{inh} = -10$, and the bias of all neurons corresponding to the RVs $\mathbf{y}$ was also set to $b_{pp} = -10$. The weights of the strong input synaptic connections to each neuron ν^{kl}, that originated from its corresponding group of neurons in $\boldsymbol{\alpha}^{kl}$ that drive it to fire, were set to $w_{pp} = 20$. During learning, the evolution of all synaptic weights was confined within a range $[w_{min}, w_{max}]$, where $w_{min} = 0$, $w_{max} = 4$ for the synapses of the $\boldsymbol{\alpha}$ neurons in the learning modules $\mathcal{N}^1$, $\mathcal{N}^2$ and $\mathcal{N}^3$, and $w_{max} = 2$ for the synapses of the $\boldsymbol{\alpha}$ neurons of module $\mathcal{N}^4$. Synaptic weight changes induced by the plasticity rule that would go below w_{min}, or above w_{max}, were clipped to w_{min}, or w_{max}, respectively.

At the beginning of each probabilistic inference simulation, the state of the network (defined by the RV values $\mathbf{y}(t)$) was initialized according to a random state drawn from a uniform distribution over all possible initial network states. This was done by injecting very short current pulses in the neurons in the populations $\boldsymbol{\nu}^k$. During the probabilistic inference simulations, evidence for the known values of the RVs was given to the network by injecting external currents in the $\boldsymbol{\nu}^k$ neurons of the known RVs. The currents were injected in exactly the same way and with exactly the same amplitudes as during the presentation of the examples in the learning process (see below).

During learning the presented examples were generated from the Bayesian network in Fig. 6B, with the values for the conditional probabilities given in Table 4. The prior probabilities $p(y^1 = 2)$ and $p(y^2 = 2)$ were both equal to 0.5.

Table 4. Values for the conditional probabilities in the Bayesian network in Fig. 6B used to generate examples for learning in Example 2.

	$p^*(y^3 = 2 \mid y^1 = 1, y^2)$	$p^*(y^3 = 2 \mid y^1 = 2, y^2)$	$p^*(y^4 = 2 \mid y^2)$
$y^2 = 1$	0.13	0.87	0.13
$y^2 = 2$	0.87	0.13	0.87

During the presentation of the examples, the neuron ν^{kl} had injected a strong positive current $I_{pp+} = 30$ if $y^k = l$ in the example, or a strong negative current $I_{pp-} = -30$ if in the example $y^k \neq l$. Additionally, if in the current example $y^k = l$, then a strong negative external current with amplitude $I_F^- = -80$ was injected in each of the neurons in the population $\boldsymbol{\alpha}^k$ that are not in the subpopulation $\boldsymbol{\alpha}^{kl}$, whereas the neurons in $\boldsymbol{\alpha}^{kl}$ did not receive any external current.

Each subpopulation $\boldsymbol{\alpha}^{kl}$ in every learning module in the network consisted of 2 neurons. All excitatory neurons in the network had an alpha shaped EPSP defined by the kernel

$$\epsilon_\alpha(t) = \begin{cases} \epsilon_0 \cdot e \cdot (\frac{t}{\tau_\alpha} + t_1) \cdot \exp(-(\frac{t}{\tau_\alpha} + t_1)) - \frac{1}{2} & \text{if } 0 < t < (t_2 - t_1)\tau_\alpha, \\ 0 & \text{otherwise.} \end{cases} \tag{81}$$

Here $\epsilon_0 = 2.8$ is a scaling factor, t_1 and t_2 and are the points in time where the basic alpha kernel of the form $e \cdot t \cdot \exp(-t)$ is equal to $\frac{1}{2}$, and $\tau_\alpha = 8.5$ms is the time constant of the alpha kernel. The same shape of the EPSPs was also used in (Pecevski et al., 2011). After an incoming spike at the synapse at time t_1^f, the time course of the EPSP is equal to $w\epsilon_\alpha(t - t_1^f)$ (w is the synaptic efficacy) until the next spike at t_2^f after which it is set to $w\epsilon_\alpha(t - t_2^f)$. The PSPs at the synapses connecting from, and to, the inhibitory neurons of the lateral inhibition had a rectangular shape with duration of $\tau = 15$ms. The rectangular IPSPs of the inhibitory neurons approximate the effect that fast-spiking bursting inhibitory neurons with short duration IPSPs would have on the membrane potential of the neurons $\boldsymbol{\alpha}$. The synaptic weights of the connections from the inhibitory neurons in the lateral inhibition to the neurons $\boldsymbol{\alpha}$ were all equal to $w_{i2e} = -7$.

Before the start of learning, the synaptic weights of all neurons in $\boldsymbol{\alpha}$ were initialized randomly from a Gaussian distribution with mean $\overline{w}_{init} = w_{max}/3$ and standard deviation $\sigma_{w0} = 0.1$. If the randomly drawn value was not in the interval $[w_{min}, w_{max}]$, then it was redrawn again until the new value was in the interval (the same type of redrawing was done for the initial values of the biases). The initial biases of the $\boldsymbol{\alpha}$ neurons were randomly drawn from a Gaussian distribution with mean $\overline{b}_{init} = 5$ and standard deviation $\sigma_{b0} = 0.1$. The evolution of the biases of the neurons in $\boldsymbol{\alpha}$ was restricted to be in the range $[b_{min}, b_{max}]$ with $b_{min} = -30$ and $b_{max} = 5$. If the intrinsic plasticity changed the bias above b_{max} or below b_{min}, the bias was clipped to b_{max} or b_{min} respectively. In the learning rules we introduced an additional scaling factor $T = 0.58$ in the potentiation part. In particular, the potentiation part of the learning rule (1) for the weights had the form $\Delta w = e^{-T(w+w_-)} - 1$, and the potentiation part of the learning rule (2) for the biases had form $\Delta b = \tau e^{-T(b+b_-)}$.

The learning process lasted 1200 seconds of biological time. During the first 600 seconds of learning, the learning rate for the synaptic weights was decreasing linearly from $\eta = 0.002$ at $t = 0$ to $\eta = 0.0006$ at $t = 600$s. In the second 600 seconds of the learning, after $t = 600$s, the synaptic plasticity was not active, i.e. the learning rate was set to $\eta = 0$. The learning rate for the biases during the first 600 seconds was constant and equal to $\eta' = 0.01$. In the second part of learning, after $t = 600$, it had another constant value equal to $\eta' = 0.02$.

The offset parameter in the synaptic plasticity rule was equal to $w_- = 2.5\log(0.2)$, whereas the offset parameter of the intrinsic plasticity in the neuron α_j^{kl} was

$$b_-^k = \frac{2.5}{T} \cdot \left(-|B(k)| \log(0.2) - \sum_{i \in \mathbf{I}^{B(k)}} \log(M(y^i) + 1) + \log(0.02) \right) \tag{82}$$

where $|B(k)|$ is the number of RVs in the Markov blanket of y^k, $\mathbf{I}^{B(k)}$ is the set of indices of the variables y^i that are in the Markov blanket of y^k, T is the scaling factor in the learning rule (see above) and $M(y^i)$ denotes the largest value of the RV y^i. As the neurons $\boldsymbol{\alpha}^k$ in different learning modules have different number of input synapses, the bias offset parameter was set for each neuron to counter-balance appropriately the average total input it receives from the input synapses.

In Fig. 7B, the frequency of the network states was calculated from the spontaneous activity of the neural network after learning, simulated for 20 seconds biological time. The neural network was initialized to be in a random network state at $t = 0$. In Fig. 8C, the KL divergence was calculated at one minute time intervals, i.e. at time points $t = 60\,i$ seconds ($i = 0, ..., 19$). At each time point the distribution $p(\mathbf{y}; \boldsymbol{\theta})$ was estimated by simulating the neural network with the values of the synaptic weights and biases of the neurons at the particular time point. The network was simulated for 1500 seconds, and the estimated probabilities were calculated from the network states in the time interval $t \in [100s, 1500s]$ of the simulation. In panels A, B and D of Fig. 8 the distributions $p^k(y^k, \mathbf{y}^{B(k)}; \boldsymbol{\theta}^k)$ and $p^k(y^k | \mathbf{y}^{B(k)}; \boldsymbol{\theta}^k)$ of the learning modules were calculated analytically by using (5) and then marginalizing $\boldsymbol{\alpha}$.

Details to the computer simulations in example 1. The target probability distribution $p^*(x^1, x^2, z)$ from which examples were generated is given in Table 5. The scaling factor T in the learning rules (see section "Details to the computer simulation in example 2") was set to $T = 0.4$. The maximum weight value was in this example $w_{max} = 5$. Before learning, the synaptic weights of all neurons in $\boldsymbol{\alpha}$ were initialized randomly from a Gaussian distribution with mean $\overline{w}_{init} = 3.0$ and standard deviation $\sigma_{w0} = 0.1$. All other parameters in this example that define the learning module as well as the learning process were the same as the parameters of the learning modules in the computer simulation in Example 2 (see section "Details to the computer simulation in example 2").

Table 5. The target probability distribution $p^*(x^1, x^2, z)$ in example 1.

	$p^*(x^1, x^2, z = 1)$	$p^*(x^1, x^2, z = 2)$
$x^1 = 1, x^2 = 1$	0.04	0.04
$x^1 = 1, x^2 = 2$	0.21	0.21
$x^1 = 2, x^2 = 1$	0.04	0.21
$x^1 = 2, x^2 = 2$	0.21	0.04

References

Abeles M, Bergman H, Gat I, Meilijson I, Seidemann E, Tishby N, Vaadia E (1995) Cortical activity flips among quasi-stationary states. *Proc Natl Acad Sci U S A* 92:8616–8620.

Ackley DH, Hinton GE, Sejnowski TJ (1985) A learning algorithm for Boltzmann machines. *Cognitive Science* 9:147 – 169.

Beck JM, Ma WJ, Kiani R, Hanks T, Churchland AK, Roitman J, Shadlen MN, Latham PE, Pouget A (2008) Probabilistic population codes for Bayesian decision making. *Neuron* 60:1142–1152.

Beck J, Heller K, Pouget A (2012) Complex inference in neural circuits with probabilistic population codes and topic models In Bartlett P, Pereira F, Burges C, Bottou L, Weinberger K, editors, *Advances in Neural Information Processing Systems 25*, pp. 3068–3076.

Beck JM, Latham PE, Pouget A (2011) Marginalization in neural circuits with divisive normalization. *The Journal of Neuroscience* 31:15310–15319.

Bengio Y, Lee D, Bornschein J, Lin Z (2015) Towards biologically plausible deep learning. *CoRR* abs/1502.04156.

Berkes P, Orbán G, Lengyel M, Fiser J (2011) Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science* 331:83–87.

Besag J (1975) Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society Series D (Statistician)* 24:179–195.

Bishop CM (2006) *Pattern Recognition and Machine Learning* Springer, New York.

Bonawitz E, Denison S, Griffiths TL, Gopnik A (2014) Probabilistic models, learning algorithms, and response variability: Sampling in cognitive development. *Trends in cognitive sciences* 18:497–500.

Brea J, Senn W, Pfister JP (2013) Matching recall and storage in sequence learning with spiking neural networks. *The Journal of Neuroscience* 33:9565–9575.

Buesing L, Bill J, Nessler B, Maass W (2011) Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput Biol* 7:e1002211.

Caroni P (2015) Inhibitory microcircuit modules in hippocampal learning. *Current Opinion in Neurobiology* 35:66–73.

Carreira-Perpinan MA, Hinton GE (2005) On contrastive divergence learning In Cowell RG, Ghahramani Z, editors, *Artificial Intelligence and Statistics*, pp. 33–41.

Chen JL, Carta S, Soldado-Matraner J, Schneider BL, Helmchen F (2013) Behaviour-dependent recruitment of long-range projection neurons in somatosensory cortex. *Nature* 499:336–340.

Cudmore RH, Turrigiano GG (2004) Long-term potentiation of intrinsic excitability in LV visual cortical neurons. *Journal of Neurophysiology* 92:341–348.

Daoudal G, Debanne D (2003) Long-term plasticity of intrinsic excitability: Learning rules and mechanisms. *Learning & Memory* 10:456–465.

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39:pp. 1–38.

Deneve S (2007) Bayesian spiking neurons II: Learning. *Neural Computation* 20:118–145.

Denison S, Bonawitz E, Gopnik A, Griffiths TL (2013) Rational variability in children's causal inferences: The sampling hypothesis. *Cognition* 126:285 – 300.

Douglas RJ, Martin KA (2004) Neuronal circuits of the neocortex. *Annual Review of Neuroscience* 27:419–451 PMID: 15217339.

Faisal AA, Selen LPJ, Wolpert DM (2008) Noise in the nervous system. *Nat Rev Neurosci* 9:292–303.

Gerstner W, Kistler W, Naud R, Paninski L (2014) *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition* Cambridge University Press.

Griffiths TL, Tenenbaum JB (2006) Optimal predictions in everyday cognition. *Psychological Science* 17:767–773.

Habenschuss S, Jonke Z, Maass W (2013a) Stochastic computations in cortical microcircuit models. *PLOS Computational Biology* 9:e1003311.

Habenschuss S, Puhr H, Maass W (2013b) Emergence of optimal decoding of population codes through STDP. *Neural Computation* 25:1371–1407.

Haeusler S, Schuch K, Maass W (2009) Motif distribution, dynamical properties, and computational performance of two data-based cortical microcircuit templates. *Journal of Physiology-Paris* 103:73–87.

Haykin SS (2009) *Neural Networks and Learning Machines (Vol. 3)* Upper Saddle River: Pearson Education.

Hinton GE (2002) Training products of experts by minimizing contrastive divergence. *Neural Computation* 14:1771–1800.

Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. *Neural Computation* 18:1527–1554.

Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences* 79:2554–2558.

Hopfield JJ (1984) Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences* 81:3088–3092.

Jank W (2006) The EM algorithm, its randomized implementation and global optimization: Some challenges and opportunities for operations research In Alt F, Fu M, Golden B, editors, *Perspectives in Operations Research*, Vol. 36 of *Operations Research/Computer Science Interfaces Series*, pp. 367–392. Springer US.

Jolivet R, Rauch A, Lüscher HR, Gerstner W (2006) Predicting spike timing of neocortical pyramidal neurons by simple threshold models. *Journal of Computational Neuroscience* 21:35–49.

Jones LM, Fontanini A, Sadacca BF, Miller P, Katz DB (2007) Natural stimuli evoke dynamic sequences of states in sensory cortical ensembles. *Proceedings of the National Academy of Sciences* 104:18772–18777.

Kampa BM, Letzkus JJ, Stuart GJ (2006) Cortical feed-forward networks for binding different streams of sensory information. *Nature neuroscience* 9:1472–1473.

Kappel D, Nessler B, Maass W (2014) STDP installs in winner-take-all circuits an online approximation to hidden Markov model learning. *PLoS Comput Biol* 10:e1003511.

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

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

Koller D, Friedman N (2009) *Probabilistic Graphical Models: Principles and Techniques* Adaptive computation and machine learning. MIT Press.

Kushner HJ, Yin G (2003) *Stochastic approximation and recursive algorithms and applications*, Vol. 35 Springer.

Landau B, Smith LB, Jones SS (1988) The importance of shape in early lexical learning. *Cognitive Development* 3:299 – 321.

Larkum ME, Nevian T, Sandler M, Polsky A, Schiller J (2009) Synaptic integration in tuft dendrites of layer 5 pyramidal neurons: A new unifying principle. *Science* 325:756–760.

Legenstein R, Maass W (2011) Branch-specific plasticity enables self-organization of nonlinear computation in single neurons. *The Journal of Neuroscience* 31:10787–10802.

Letzkus JJ, Wolff SBE, Lthi A (2015) Disinhibition, a circuit mechanism for associative learning and memory. *Neuron* 88:264–276.

Litvak S, Ullman S (2009) Cortical circuitry implementing graphical models. *Neural Computation* 21:3010–3056.

Ma WJ, Beck JM, Latham PE, Pouget A (2006) Bayesian inference with probabilistic population codes. *Nat Neurosci* 9:1432–1438.

Mante V, Sussillo D, Shenoy KV, Newsome WT (2013) Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* 503:78–84.

Mensi S, Naud R, Gerstner W (2011) From stochastic nonlinear integrate-and-fire to generalized linear models In Shawe-Taylor J, Zemel R, Bartlett P, Pereira F, Weinberger K, editors, *Advances in Neural Information Processing Systems 24*, pp. 1377–1385.

Mozzachiodi R, Byrne JH (2010) More than synaptic plasticity: role of nonsynaptic plasticity in learning and memory. *Trends Neurosci* 33:17–26.

Murphy KP (2012) *Machine Learning: a Probabilistic Perspective* MIT press.

Neal RM (1992) Connectionist learning of belief networks. *Artificial Intelligence* 56:71 – 113.

Nessler B, Pfeiffer M, Buesing L, Maass W (2013) Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput Biol* 9:e1003037.

Pecevski D, Buesing L, Maass W (2011) Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons. *PLoS Comput Biol* 7:e1002294.

Pecevski D, Kappel D, Jonke Z (2014) NEVESIM: Event-driven neural simulation framework with a Python interface. *Frontiers in Neuroinformatics* 8:fninf.2014.00070.

Pecevski D, Natschlger T, Schuch K (2009) PCSIM: a parallel simulation environment for neural circuits fully integrated with Python. *Frontiers in Neuroinformatics* 3:neuro.11.011.2009.

Pfister JP, Toyoizumi T, Barber D, Gerstner W (2006) Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural Computation* 18:1318–1348.

Rao RPN (2006) *Bayesian Brain: Probabilistic Approaches to Neural Coding*, chapter Neural Models of Bayesian Belief Propagation, pp. 235–264 MIT Press.

Rezende DJ, Wierstra D, Gerstner W (2011) Variational learning for recurrent spiking networks In Shawe-Taylor J, Zemel R, Bartlett P, Pereira F, Weinberger K, editors, *Advances in Neural Information Processing Systems 24*, pp. 136–144.

Rigotti M, Barak O, Warden MR, Wang XJ, Daw ND, Miller EK, Fusi S (2013) The importance of mixed selectivity in complex cognitive tasks. *Nature* 497:585–590.

Salakhutdinov R, Hinton G (2012) An Efficient Learning Procedure for Deep Boltzmann Machines. *Neural Computation* 24:1967–2006.

Schacter D (2002) *The Seven Sins of Memory: How the Mind Forgets and Remembers* Houghton Mifflin Harcourt.

Siegelmann HT, Holzman LE (2010) Neuronal integration of dynamic sources: Bayesian learning and bayesian inference. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 20:037112.

Sjöström PJ, Turrigiano GG, Nelson SB (2001) Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron* 32:1149 – 1164.

Song S, Sjöström. PJ, Reigl M, Nelson S, Chklovskii DB (2005) Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS Biol* 3:e68.

Steimer A, Maass W, Douglas R (2009) Belief propagation in networks of spiking neurons. *Neural Computation* 21:2502–2523.

Steinbuch K (1961) Die Lernmatrix. *Kybernetik* 1:36–45.

Stickgold R, Walker MP (2013) Sleep-dependent memory triage: Evolving generalization through selective processing. *Nat Neurosci* 16:139–145.

Toyoizumi T, Pfister JP, Aihara K, Gerstner W (2005) Generalized Bienenstock-Cooper-Munro rule for spiking neurons that maximizes information transmission. *Proceedings of the National Academy of Sciences* 102:5239–5244.

Vul E, Pashler H (2008) Measuring the crowd within: Probabilistic representations within individuals. *Psychological Science* 19:645–647.

Wei GCG, Tanner MA (1990) A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association* 85:699–704.

Williams SR, Stuart GJ (2002) Dependence of EPSP efficacy on synapse location in neocortical pyramidal neurons. *Science* 295:1907–1910.

Yoshimura Y, Dantzker JLM, Callaway EM (2005) Excitatory cortical neurons form fine-scale functional networks. *Nature* 433:868–873.

Figures

Figure 1. STDP curves of the synaptic plasticity rule. **A)** The STDP curves show the weight change for a presynaptic spike at time t_{pre} and a postsynaptic spike at time t_{post}, for different time differences $t_{post} - t_{pre}$. The red curve represents STDP for the simple rule that is used in the theoretical derivations. In computer simulations we used also an STDP rule shown with the blue curve, that has a smoother, more biologically realistic shape. **B)** The change of the synaptic efficacy after a stimulation protocol where both the presynaptic and postsynaptic neuron fire at a frequency of $20Hz$, for different time differences Δt between a postsynaptic and presynaptic spike. The STDP curve shifts more towards LTP, and depression is no longer time independent due to overlapping PSPs (see Fig. 4 in (Nessler et al., 2013) for details). This STDP curve is quite similar to experimental data (see e.g. (Sjöström et al., 2001)).

Figure 2. Structure of a stochastic association module, that is able to learn probabilistic associations between multinomial variables $\mathbf{x} = (x^1, \ldots, x^I)$ and z through STDP. Populations of neurons $\boldsymbol{\chi}^i$ $(i = 1, \ldots, I)$ on the first layer encode the values of input variables x^i. The population of neurons ζ on the third layer encodes the value of z. The hidden layer consists of populations of excitatory neurons $\boldsymbol{\alpha}^l$ $(l = 1, \ldots, L)$ that are subject to lateral inhibition. STDP applied to the weights $w^l_{im,j}$ of synaptic connections from the first layer to the neurons $\boldsymbol{\alpha}$ on the hidden layer enables the network to approximate for any network input $\mathbf{x}$ through the firing probability of neurons on the 3^{rd} layer the distribution of values z that were associated with $\mathbf{x}$ in previously processed examples $\langle \mathbf{x}, z \rangle$.

Figure 3. Learning results for Example 1. A) Structure of the learning module. There are two subpopulations $\boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2$ of hidden neurons that both receive inputs from the two populations on layer 1 that encode the input variables x^1 and x^2. Each subpopulation of hidden neurons projects to a different neuron in the population coding of the variable z on layer 3. **B)** Plots on the left: The target probability distribution $p^*(\mathbf{x}, z)$ of the examples (grey bars) and the internal model $p(\mathbf{x}, z; \boldsymbol{\theta})$ (blue bars) that is extracted from the examples by the hidden neurons. The learned probabilities match the target probabilities quite well. On the right: the two mixture components $p(\mathbf{x}, z = 2, \alpha^2_1 \text{ fires } ; \boldsymbol{\theta})$ and $p(\mathbf{x}, z = 2, \alpha^2_2 \text{ fires } ; \boldsymbol{\theta})$ represented by the hidden neurons α^2_1 and α^2_2 in the subpopulation $\boldsymbol{\alpha}^2$. Each has specialized to represent one of the two modes. The resulting internal model $p(\mathbf{x}, z = 2; \boldsymbol{\theta})$ is a sum of these two mixture components. **C)** Same as the plots on the left in panel B, but for a larger network where 4 hidden neurons were used in each subpopulation $\boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2$ of hidden neurons. This larger size of the subpopulations is suggested by the network construction from (Pecevski et al., 2011), since the vector $\mathbf{x}$ can assume four different values. But a comparison with panel C shows that smaller subpopulations suffice here for good learning performance. **D)** Plots on the left: the target probabilities $p^*(z|\mathbf{x})$ (grey bars) compared with the learned firing probabilities of the output neurons ζ^1 and ζ^2 that represent $p(z = 1|\mathbf{x}; \boldsymbol{\theta})$ and $p(z = 2|\mathbf{x}; \boldsymbol{\theta})$ respectively. Plots on the right: the probability of firing in response to different inputs $\mathbf{x}$ for the two hidden neurons α^2_1 and α^2_2 that drive ζ^2 to fire. **E)** Same as the plots on the left in panel D, but for the larger network as in C. Again, one sees that fewer hidden neurons are needed here than in the construction of (Pecevski et al., 2011). **F)** Firing activity of the hidden neurons and output neurons in the module in response to two different input patterns $(x^1, x^2) = (1, 1)$ and $(x^1, x^2) = (2, 1)$. The firing rates of α^2_1 and α^2_2 correspond to their probabilities of firing shown in panel D.

Figure 4. Recursive combination of learning modules. The learning module for the RV y^k at the bottom has the same structure as the modules shown in Fig. 2 and 3. For learning complex distributions p^* its input variables $x^1, \ldots, x^l$ form a Markov blanket of y^k. Each variable x^i is encoded by the same population coding as the output variables of learning modules, and can therefore be produced by the output of another learning module (as shown for the RV x^l). As here y^k is in the Markov blanket of x^l, y^k appears among the input variables of the upper module, and its corresponding input neurons are the same as the output neurons of the lower module.

Figure 5. Schematic description of the learning approach. Sequence of examples $\tilde{\mathbf{y}}(0), \tilde{\mathbf{y}}(1), \ldots$ drawn from the target distribution $p^*(\mathbf{y})$ are presented to the neural network $\mathcal{N}$. The neural network is composed of learning modules $\mathcal{N}^k$, one for each RV y^k. $\mathcal{N}^k$ learns from the components $\langle \tilde{y}^k(n), \tilde{\mathbf{y}}^{B(k)}(n) \rangle$ of examples $\tilde{\mathbf{y}}(n)$ an approximation $p^k(y^k, \mathbf{y}^{B(k)}; \boldsymbol{\theta}^k)$ of $p^*(y^k, \mathbf{y}^{B(k)})$ as indicated in Fig. 2, 3. The theory based on Expectation Maximization ensures that the total network $\mathcal{N}$ learns in this way an approximation $p(\mathbf{y}; \boldsymbol{\theta})$ of $p^*(\mathbf{y})$.

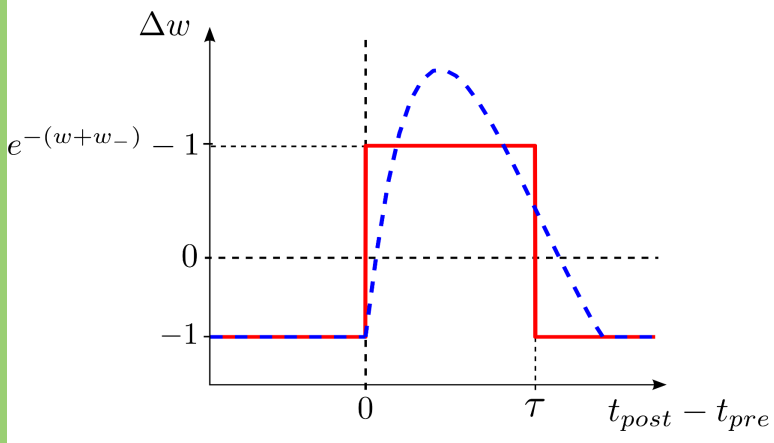
Figure 6. Description of the perceptual explaining away example. A) The two visual stimuli used in the experiment from (Knill and Kersten, 1991). Both surfaces, the upper and the lower, have identical shading profiles in the horizontal direction. Nevertheless, subjects perceive that the reflectances of the two halves of the lower panel are the same, whereas they perceive the left half of the upper panel as being darker than the right half. The different contours of the two panels suggest different 3D shapes (flat versus cylindrical), which influences subjects' perception of the reflectance of the two halves of each surface. **B)** The "explaining away" Bayesian network proposed in (Kersten and Yuille, 2003) that models the effect from panel A. It consists of 4 RVs y^1, y^2, y^3 and y^4. The relative reflectance y^1 of the surfaces can have two values: $y^1 = 2$ for different and $y^1 = 1$ for the same reflectance of the two parts of the surface. The 3D shape of the surfaces (y^2) is either cylindrical ($y^2 = 2$) or rectangular ($y^2 = 1$). The relative reflectance and the 3D shape are direct causes of the shading or the luminance change of the surfaces (y^3), which can have the profile like in the bottom part of panel B ($y^3 = 2$) or a different one ($y^3 = 1$). The 3D shape of an object causes different 2D contours (y^4), which can be either straight ($y^4 = 1$) or curved ($y^4 = 2$). The observed variables are the contour (y^4) and the shading (y^3) of the surfaces. Subjects infer the value of the relative reflectance (y^1) and the 3D shape (y^2) based on these observed cues. **C)** The structure of the neural network $\mathcal{N}$ that corresponds to the Bayesian network in panel B. For each RV y^k in the Bayesian network there is a learning module $\mathcal{N}^k$ composed of a population of neurons that outputs y^k in population coding, and a population of hidden neurons $\boldsymbol{\alpha}^k$. The learning modules are inter-connected according to the Markov blankets of the RVs in the Bayesian network as indicated in Fig. 4. For example, the RVs in the Markov blanket of y^1 are y^2 and y^3, and therefore the learning module $\mathcal{N}^1$ receives connections from $\mathcal{N}^2$ and $\mathcal{N}^3$. **D)** Structure of the learning module $\mathcal{N}^1$ for the RV y^1 in the neural network in panel C.

Figure 7. Analysis of the stationary distribution of network states of the network from Fig. 6C after learning. A) Random sample of spontaneous activity of the network (without any external input). The network switches stochastically between different network states (some of which are labeled by colors), in a manner that is qualitatively similar to experimental data from networks of neurons in the cortex (see e.g. (Abeles et al., 1995; Jones et al., 2007)). The spike trains of neurons that encode a problem variable y^k through population coding are underlined by blue lines. Spike trains immediately above are from the hidden neurons $\boldsymbol{\alpha}^k$ that drive this neuron to fire (as in Fig. 2). **B)** Frequency of network states that encode particular assignments to the problem variables y^k (shown on the x-axis) resulting from the learned stationary distribution (spontaneous activity) $p(\mathbf{y};\boldsymbol{\theta})$ of the network are shown as green bars. The coloring of the network states labels is the same as for corresponding network states in A. Black bars indicate the probabilities of the same value assignments under the distribution p^* that had produced the examples for learning.

Figure 8. Performance of the recursive combination of learning modules from Fig. 6C in learning the perceptual inference task of (Knill and Kersten, 1991) from examples. A) Evolution of the KL divergence between the target distribution $p^*(y^k, \mathbf{y}^{B(k)})$ of the examples (denoted in the plot as p_k^*) and the distribution of the internal model $p^k(y^k, \mathbf{y}^{B(k)};\boldsymbol{\theta}^k)$ during learning, for each of the learning modules $\mathcal{N}^k$ in the network ($k = 1, 2, 3, 4$). **B)** Evolution of the sum of the KL divergences shown in panel A. **C)** Evolution of the KL divergence between the internal model distribution $p(\mathbf{y};\boldsymbol{\theta})$ represented by the whole network and the target distribution $p^*(\mathbf{y})$ of the examples during learning. **D)** The plots show the target conditional distributions $p^*(y^k|\mathbf{y}^{B(k)})$ (black bars), and the learned conditional distributions $p^k(y^k|\mathbf{y}^{B(k)};\boldsymbol{\theta}^k)$ (green bars), for each problem variable y^k. The bit string labels on the x-axis denote the assignment of values to the problem variables on which each distribution is conditioned.

Figure 9. Demonstration that the network $\mathcal{N}$ from Fig. 6C has learnt explaining away. A, B) Results of probabilistic inference (estimate of posterior marginal probabilities) by the network through observation of the firing rates of neurons that encode y^1 and y^2 (while other neurons are "clamped" to encode the evidence $\mathbf{e}$) are shown as green bars. The results are from a single run with duration 400ms per evidence value. Black bars indicate corresponding values for p^* ("ground truth"). The network estimates the probabilities $p(y^1 = 2|\mathbf{e};\boldsymbol{\theta})$ (indicating different relative reflectance according to Fig. 6B) and $p(y^2 = 2|\mathbf{e};\boldsymbol{\theta})$ (indicating a cylindrical 3D shape) for two different evidence values $\mathbf{e} = (y^3, y^4)$ through sampling from its stationary distribution of network states. In the first inference we assume $\mathbf{e} = (2, 2)$ (indicating an observation with discontinuous shading of the object and curved contour), and in the second $\mathbf{e} = (2, 1)$ (observation of discontinuous shading and straight contour). **C)** The spiking activity of the network $\mathcal{N}$ during online inference of hidden causes, where the evidence $\mathbf{e}$ was switched after $t = 3$s (red vertical line) from $\mathbf{e} = (2, 2)$ to $\mathbf{e} = (2, 1)$. Spike trains of neurons from population codes for problem variables y^k are underlined by blue lines, like in Fig. 7A. **D)** Firing rates (estimated with a sliding alpha kernel $K(t) = \frac{t}{\tau}\exp(\frac{t}{\tau})$, $t = 0.1$s) of the neurons encoding $y^1 = 2$ (different relative reflectance, dashed line) and $y^2 = 2$ (cylindrical 3D shape) during the inference simulation run shown in C). The "explaining away" effect is clearly visible from the complementary evolution of the firing rates of the two neurons that represent the two potential hidden causes "cylindrical 3D shape" and "different relative reflectance".

A



B

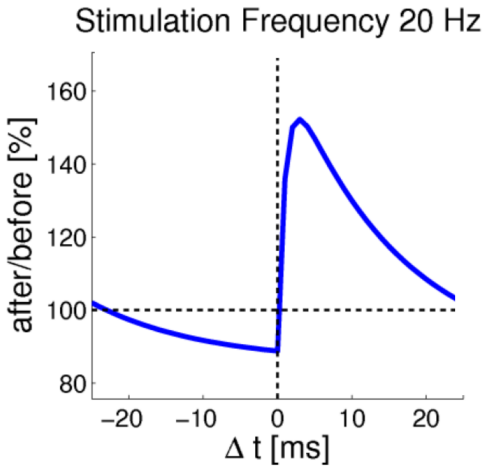


Figure 1.

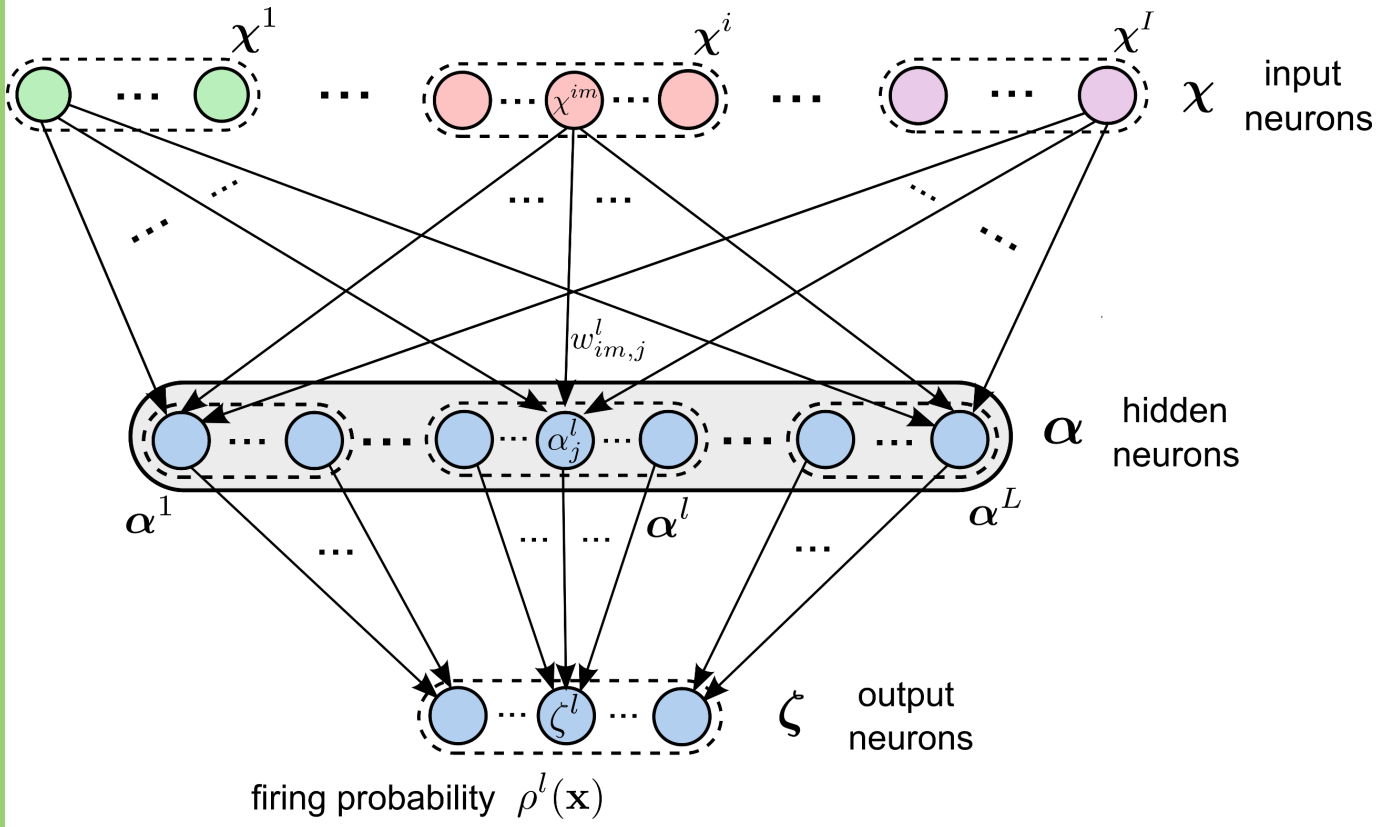


Figure 2.

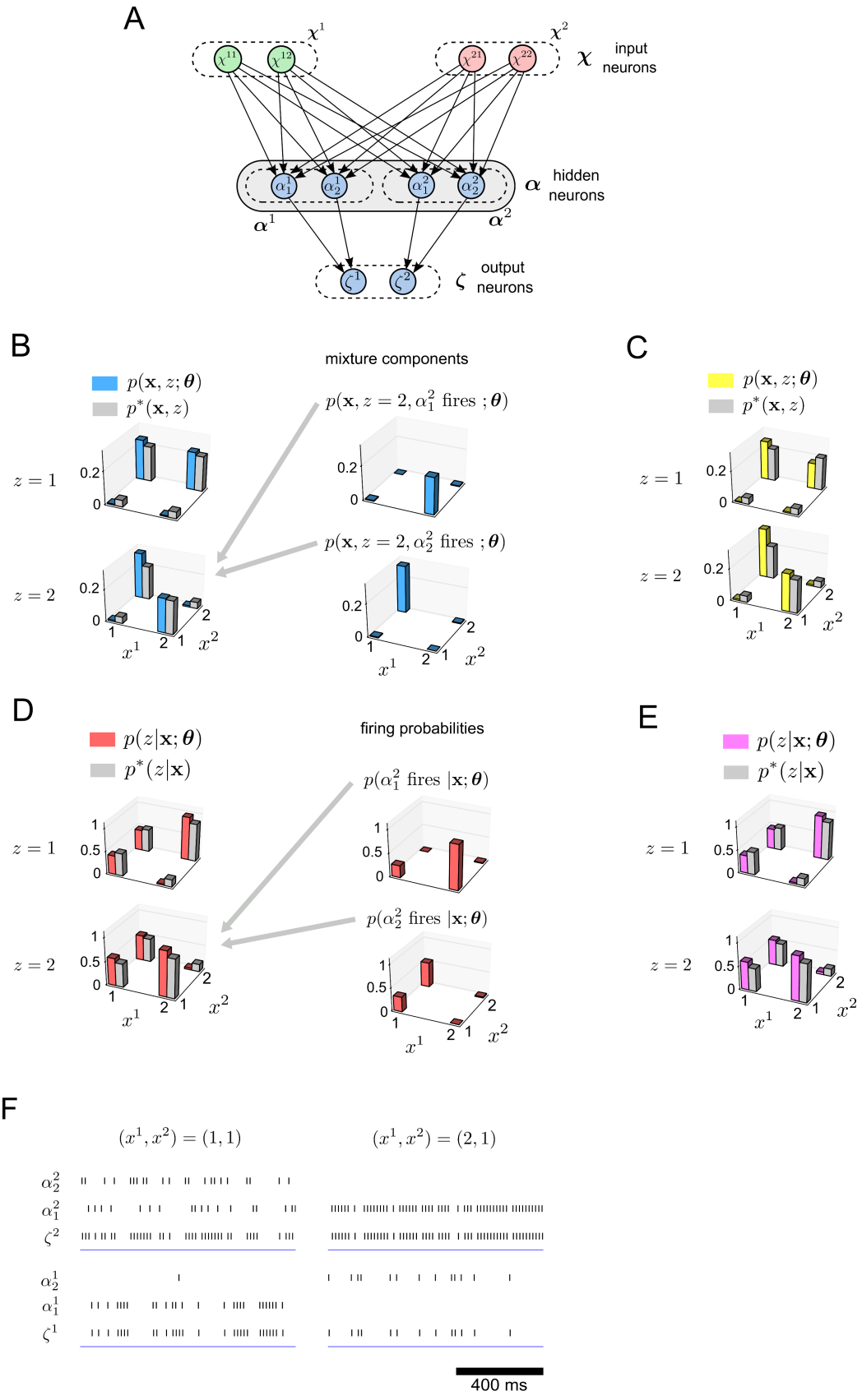


Figure 3.

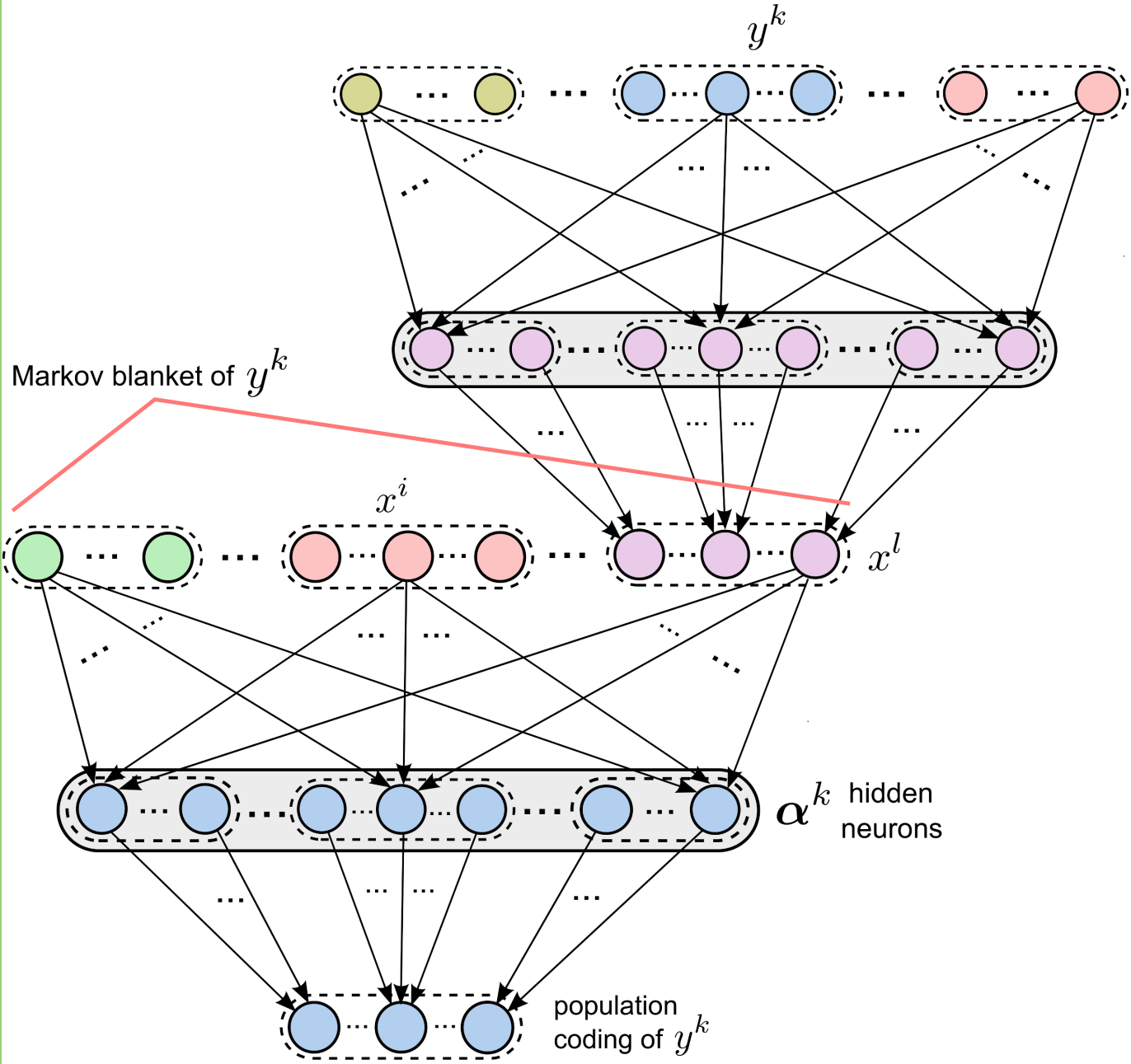


Figure 4.

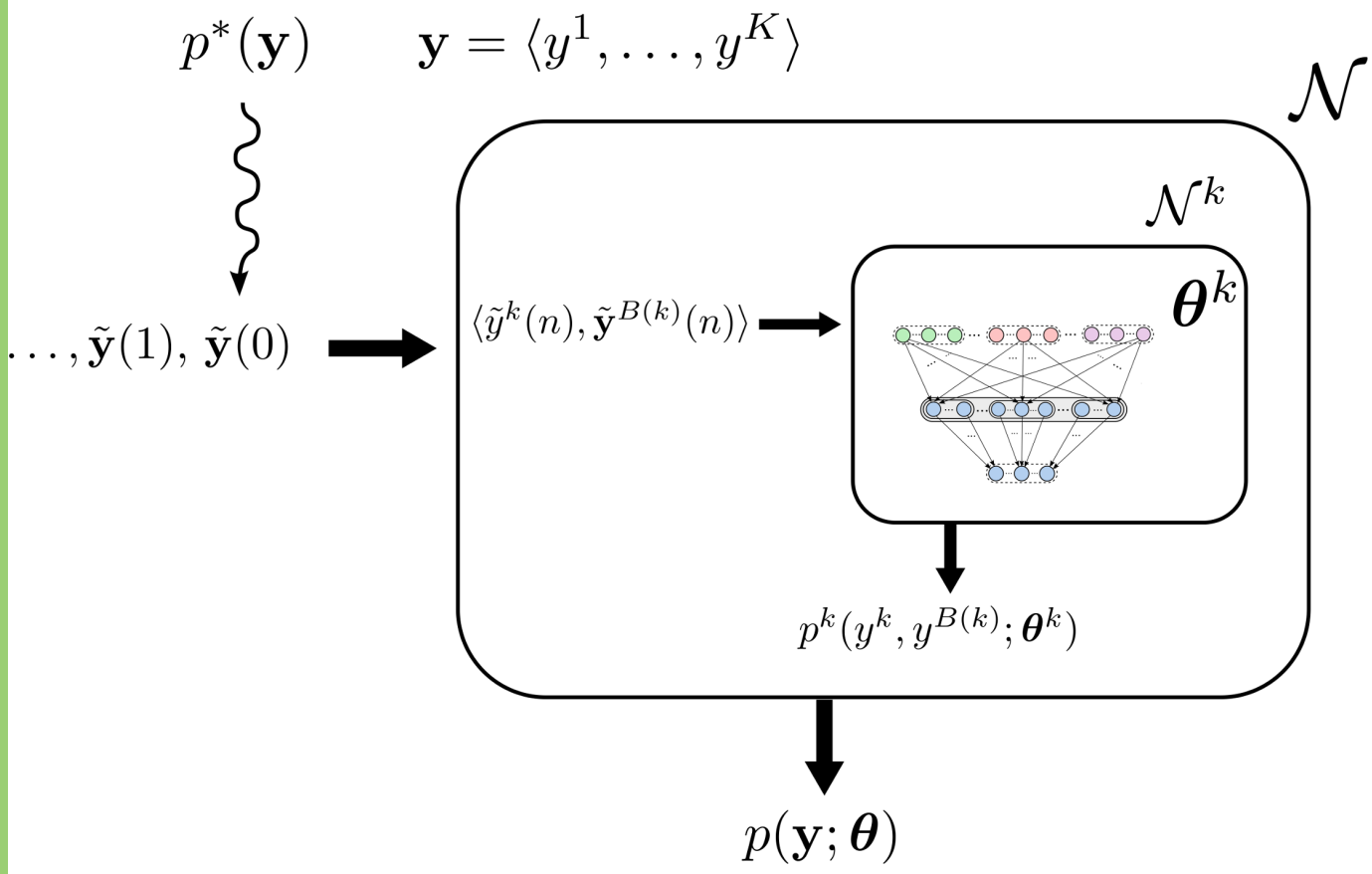


Figure 5.

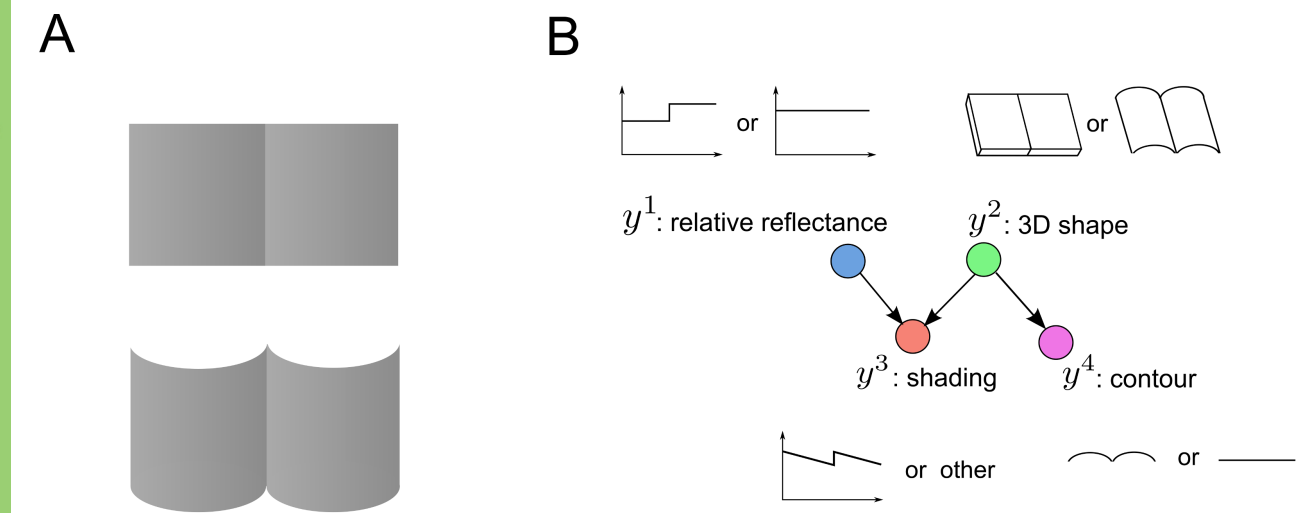
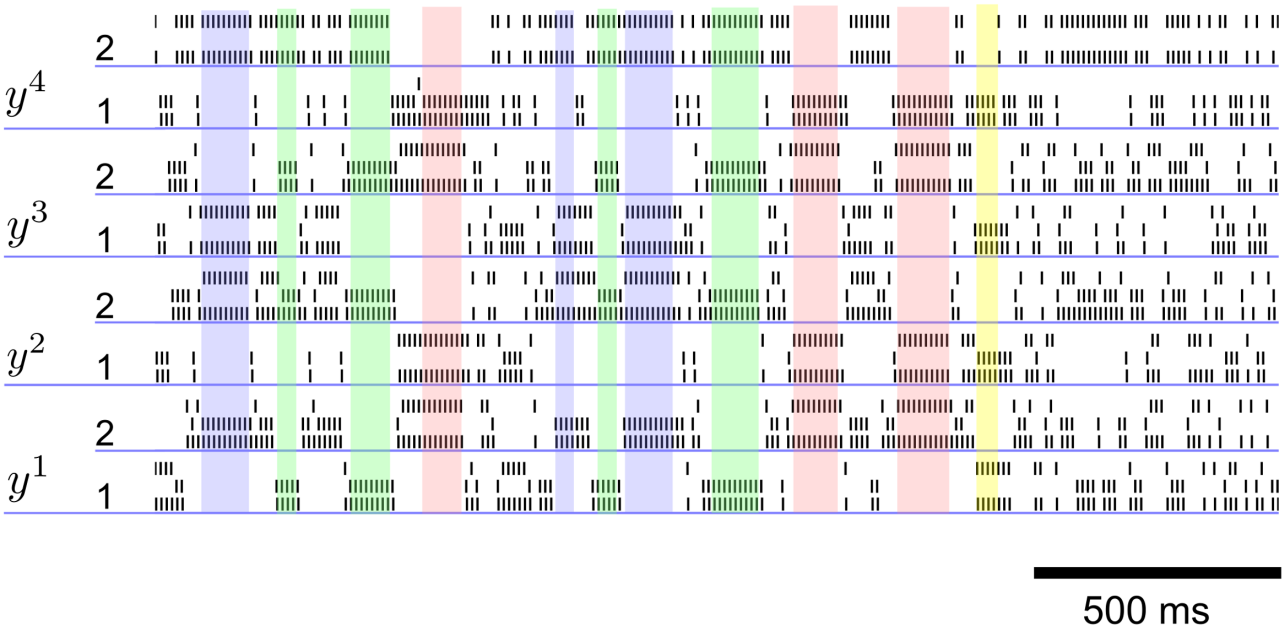


Figure 6.

A



B

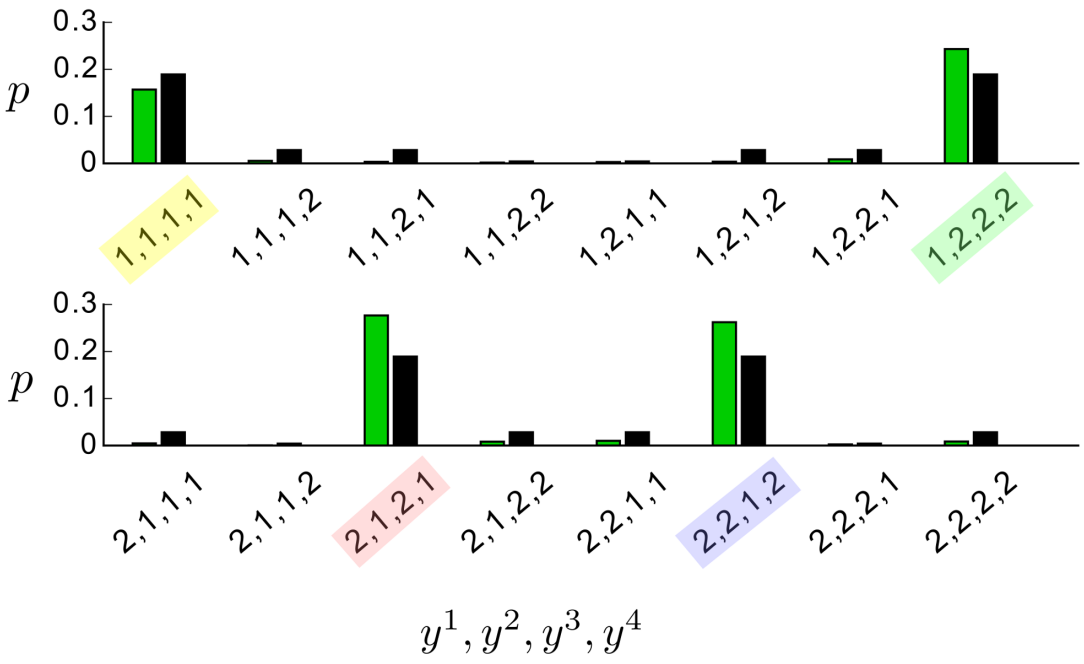


Figure 7.

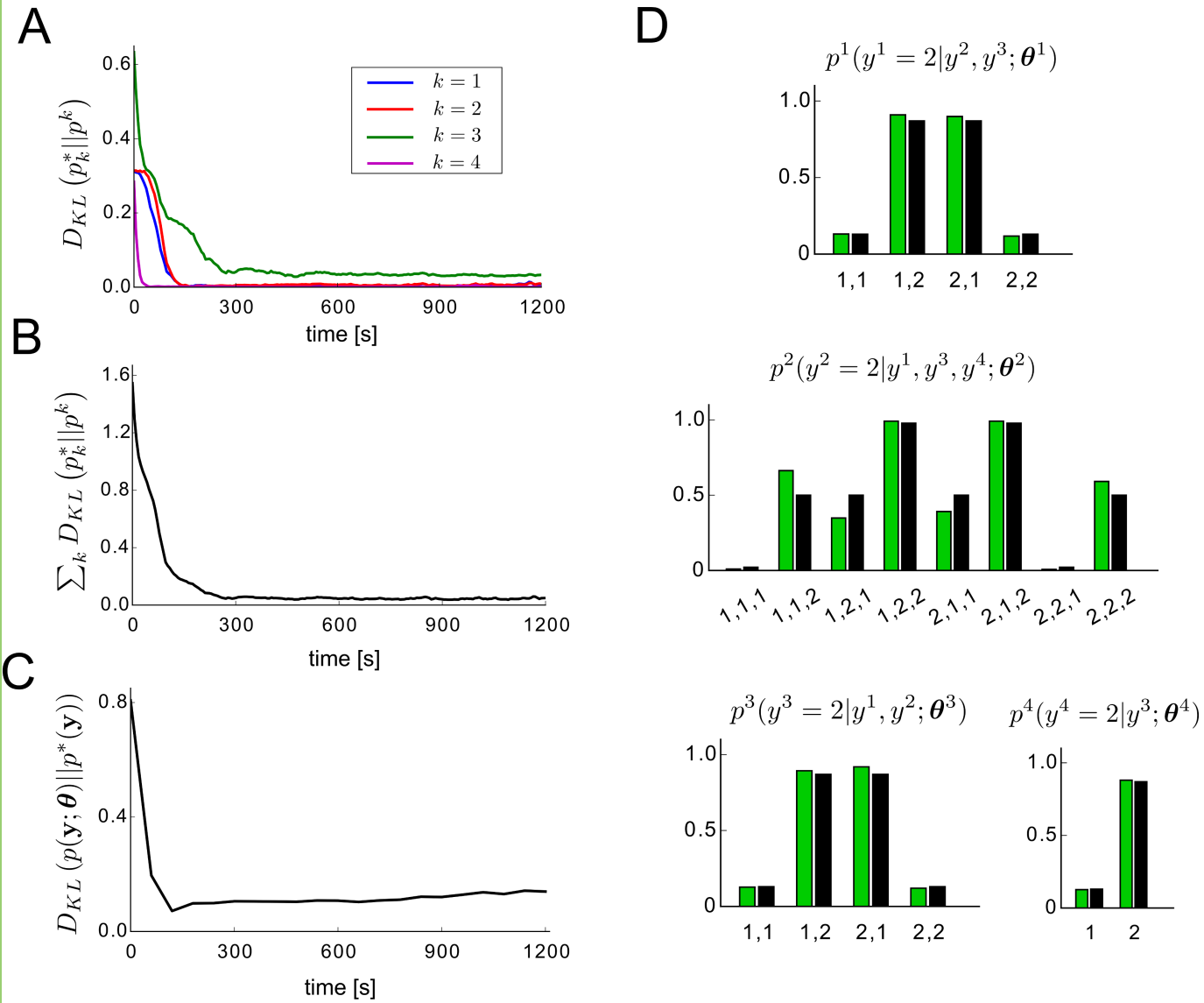


Figure 8.

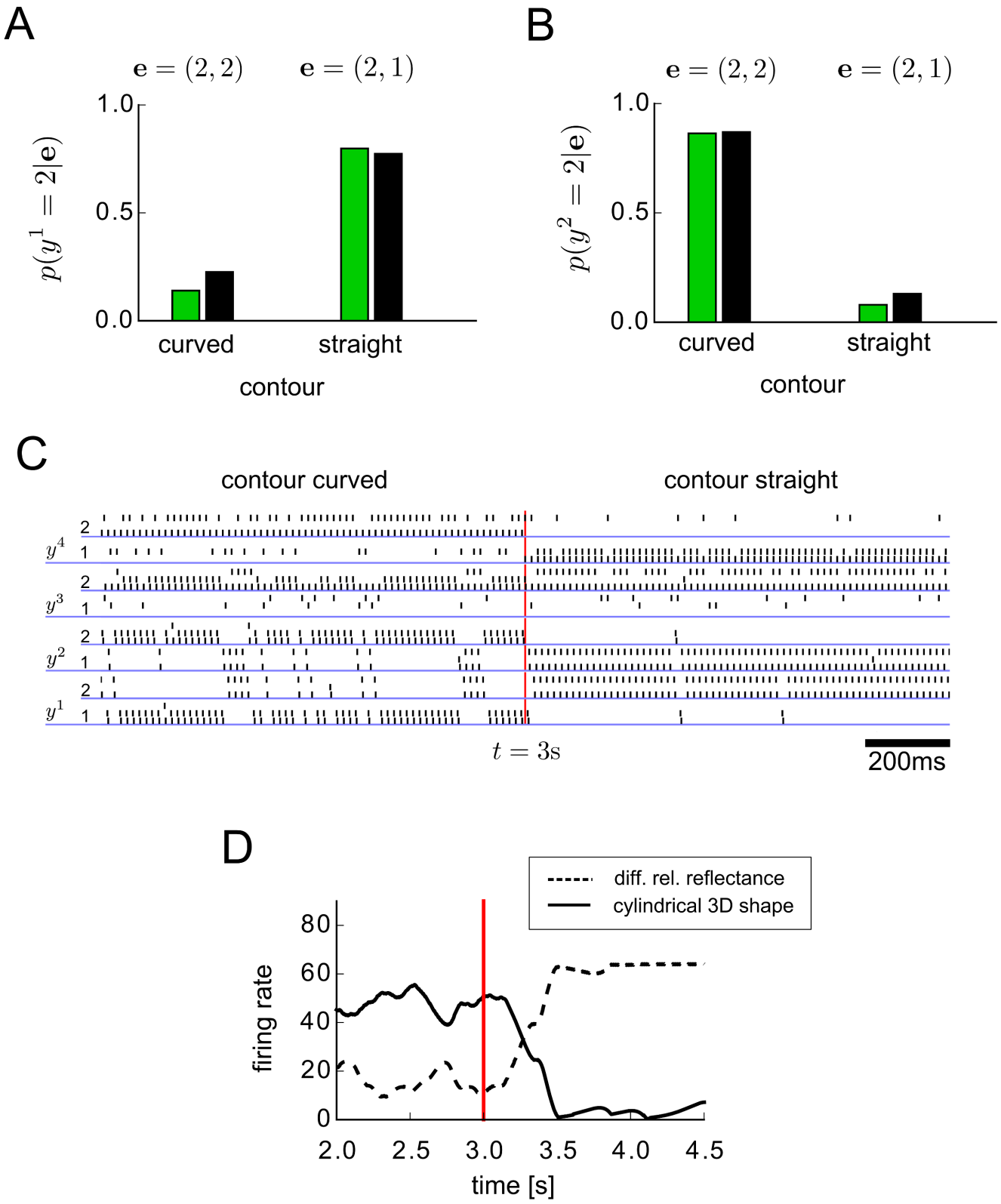


Figure 9.