Generating neural circuits that implement probabilistic reasoning

M. J. Barber*
Universidade da Madeira, Centro de Ciências Matemáticas, Campus Universitário da Penteada, 9000-390 Funchal, Portugal

J. W. Clark
Department of Physics, Washington University, Saint Louis, Missouri 63130, USA

C. H. Anderson
Department of Anatomy and Neurobiology, Washington University School of Medicine, Saint Louis, Missouri 63110, USA
(Received 17 January 2003; published 21 October 2003)

We extend the hypothesis that neuronal populations represent and process analog variables in terms of probability density functions (PDFs). Aided by an intermediate representation of the probability density based on orthogonal functions spanning an underlying low-dimensional function space, it is shown how neural circuits may be generated from Bayesian belief networks. The ideas and the formalism of this PDF approach are illustrated and tested with several elementary examples, and in particular through a problem in which model-driven top-down information flow influences the processing of bottom-up sensory input.

DOI: 10.1103/PhysRevE.68.041912 PACS number(s): 87.18.Sn, 87.19.La

I. INTRODUCTION

A. Fundamental hypothesis

In this work we elaborate upon the proposition [1] that neural populations encode and process information about analog variables in the form of probability density functions (PDFs). As demonstrated elsewhere [2], explicit representation of probabilistic descriptors of the state of knowledge of physically relevant variables subserves a powerful strategy for modeling neural circuits. By exploiting mathematical tools developed within the theory of Bayesian inference, we can establish general procedures for building and understanding models of cortical circuits that carry out well-posed information-processing tasks.

Systems based on probabilistic frameworks provide a number of strong conceptual advantages. Importantly, a neural assembly encoding the joint probability density over relevant analog variables can in principle answer any probabilistic question about these variables. Further, by implementing the Bayesian rules of inference, a probabilistic formulation provides a direct and consistent means to deal with changing sources of evidence in the system.

Motivated by these facts, we shall now extend the PDF hypothesis by devising methods for embedding joint probabilities into neural networks. This will enable us to construct neural circuit models that pool multiple sources of evidence, such as sensory inputs and any evolutionarily determined priors on the joint distribution. More specifically, we will focus on developing neural networks that use ''bottom-up'' sensory inputs to build an internal model of the data, which in turn uses ''top-down'' signals to impose global regularities on the sensory data. In the resulting neural networks, there will naturally arise distinct feedforward, feedback, and lateral connections, analogous to the neural pathways observed in the anatomy of the cerebral cortex [3].

Our treatment is based on Bayesian belief networks [4,5], graphical representations of probabilistic models that provide an efficient means for organizing the relations between the random variables of a given model. The resulting neural networks will have several properties of Bayesian belief networks, as well as more typical neural-network properties, so we will call them *neural belief networks*. Bayesian belief network have been previously utilized in the genesis of certain neural networks [6,7], although with different methods for generating the neural-network architecture and dynamics.

B. Three levels of representation

In the context of the PDF hypothesis [8], we assert that a physical variable x is described by a neural population at time t in terms of a PDF $\rho(x;t)$, rather than as a single-valued estimate $x(t)$. In general, we consider a PDF described at time t in terms of a set of D time-dependent parameters $\{A_\mu\}$. One possible description is suggested by linear decoding rules, which have been shown to be adequate in some situations [9,10], leading to PDFs represented as

$$\rho(x;t)=\sum_{\mu=1}^{D} A_\mu(t)\Phi_\mu(x). \tag{1}$$

The basis functions $\Phi_\mu(x)$ are orthonormal functions which serve to define the PDFs that the neural circuit can represent, but the amplitudes $A_\mu(t)$ cannot be interpreted as neuronal firing rates due to their arbitrarily high precision and their ability to take on negative values. Therefore, we introduce an additional representation of the PDF in terms of firing rates $a_i(t)$ and decoding functions $\phi_i(x)$ assigned to N neurons, so that

$$\rho(x;t)=\sum_{i=1}^{N} a_i(t)\phi_i(x). \tag{2}$$

Unlike the basis functions $\Phi_\mu(x)$, the decoding functions $\phi_i(x)$ form a highly redundant, overcomplete representation ($N \gg D$) suitable for use with neuronal units having biologically realistic precision (typically some 2–3 bits [11]).

*Electronic address: mjb@uma.pt

The abstract representation defined by Eq. (1) will underlie the representation in the neuron space defined by Eq. (2). This allows us to deal with the issue of how PDFs can be precisely implemented in populations of neurons by focusing on the mapping between the minimal space and the space of neurons [2]. Thus, neural belief networks can be developed in the theoretically convenient abstract representation, and then be implemented in more realistic networks of low-precision model neurons. Adopting the terminology of Zemel *et al.* [12], we denote the set of physical variables as the *implicit space* and the measurable quantities as the *explicit space*. Extending their nomenclature, we denote the abstract space of Eq. (1) as the *minimal space*. The explicit space of neurons constitutes a biological implementation of the desired computations in the implicit space, while the minimal space, whose properties are more conducive to formal analysis, provides a valuable bridge between the two other spaces.

We have developed rules to transform between representations in the three spaces [2]. Equations (1) and (2) respectively define how to transform representations in the minimal and explicit spaces into a representation (PDF) in the implicit space. With the remaining rules (summarized below), we can readily switch between the three representations and approach any task in the most appropriate space.

For the orthonormal basis $\{\Phi_\mu(x)\}$, the minimal space coefficients $A_\mu(t)$ are found using the encoding rule

$$A_\mu(t)=\int \Phi_\mu(x)\rho(x;t)dx. \tag{3}$$

The coefficients in the explicit space, i.e., the neural firing rates $a_i(t)$, cannot be found in this direct fashion. We utilize a set of encoding functions $\hat{\phi}_i(x)$ and an encoding rule of the form

$$a_i(t)=f\left(\int \hat{\phi}_i(x)\rho(x;t)dx\right). \tag{4}$$

This explicit-space encoding rule is patterned after the neural responses associated with the population vector of Georgopoulos *et al.* [9]. The activation function f in general is nonlinear, and serves to prevent negative firing rates. One possible choice for f is rectification, i.e., $f(x)=x$ for $x>0$ and $f(x)=0$ otherwise; we make use of rectifying activation functions throughout this work.

To relate the explicit and minimal spaces, we express the encoding and decoding functions in terms of the orthonormal basis, so that

$$\phi_i(x)=\sum_{\nu=1}^{D} \kappa_{\nu i}\Phi_\nu(x), \tag{5}$$

$$\hat{\phi}_i(x)=\sum_{\nu=1}^{D} \hat{\kappa}_{i\nu}\Phi_\nu(x). \tag{6}$$

It can be shown that the transformation coefficients $\kappa_{\nu i}$ and $\hat{\kappa}_{i\nu}$ also relate the $A_\mu(t)$ and $a_i(t)$, such that

$$A_\nu(t)=\sum_{i=1}^{N} \kappa_{\nu i}a_i(t), \tag{7}$$

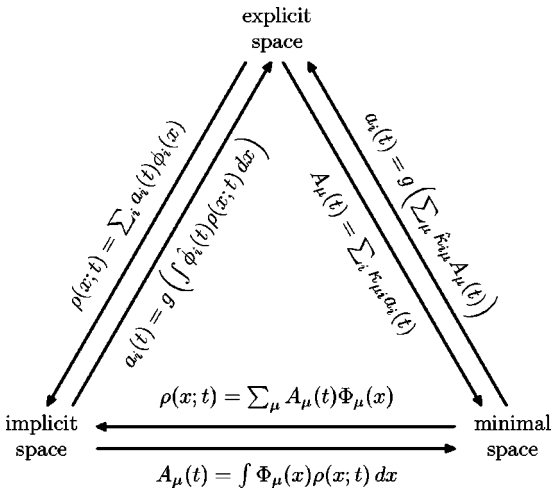

FIG. 1. Formal transformations between the explicit, implicit, and minimal spaces.

$$a_i(t)=f\left(\sum_{\nu=1}^{D} \hat{\kappa}_{i\nu}A_\nu(t)\right). \tag{8}$$

The formal relations between the three spaces are illustrated in Fig. 1.

C. Obtaining the representation

Using Eqs. (2)–(8), we can transform between representations in the explicit, implicit, and minimal spaces using the encoding functions $\hat{\phi}_{i(x)}$, the decoding functions $\phi_i(x)$, and the associated transformation coefficients $\kappa_{\nu i}$ and $\hat{\kappa}_{i\nu}$. We present here a brief discussion of how to determine these functions and coefficients. Methods for their explication have been developed and presented in greater detail in a previous paper [2], which includes a discussion of the effect of the dimensionality D of the minimal space. The method we present here is not unique, and could be modified, e.g., by using a different error measure than that in Eq. (9) to derive the decoders.

Consider a known "target" PDF $\tilde{\rho}(x|\xi_1,\xi_2,\ldots,\xi_M)$ described in terms of M time-dependent parameters ξ_i. These parameters represent one or more behaviorally relevant quantities, e.g., eye position, properties of a visual scene, or limb position. We can represent the target PDF in the minimal space using the orthonormal basis $\{\Phi_\nu(x);\nu=1,2,\ldots,D\}$ in a straightforward fashion.

In the explicit space, we define a target set of neural firing rate response profiles $\{\tilde{a}_i(\xi_1,\xi_2,\ldots,\xi_M):i=1,2,\ldots,N\}$. For determination of the representation, neural noise and limited firing rate precision can be effectively treated by adding a noise source ε_i to the target firing rates $\tilde{a}_i$.

The decoders are constructed such that the decoded PDF defined by Eq. (2) matches well with the target PDF. We introduce a quadratic error measure

$$E_{\mathrm{dec}}=\left\langle \int D_{\mathrm{dec}}^2(x,\{\xi_\nu\})\rho(\{\xi_\nu\})dx\,d\xi_1 d\xi_2\cdots d\xi_M \right\rangle_{\{\varepsilon_i\}} \tag{9}$$

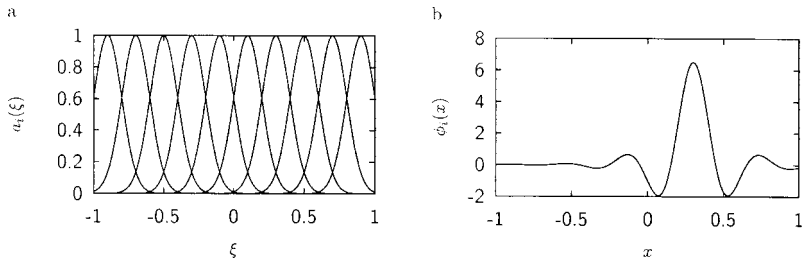

FIG. 2. Gaussian structured representation. (a) Gaussian functions defining the firing rate responses of the neurons. Orthonormal bases for the minimal space are formed from these functions. The encoders are identical in shape to the firing rate profiles. (b) Typical decoder for the Gaussian neural responses. This decoder corresponds to the neural response profile centered at 0.3 (same as the decoder). Other decoders are of essentially the same shape, but centered at the corresponding firing rate profile.

on the deviations

$$D_{\mathrm{dec}}(x,\{\xi_\nu\})=\tilde{\rho}(x|\{\xi_\nu\})-\sum_{i=1}^{N}\left[\tilde{a}_i(\{\xi_\nu\})+\varepsilon_i\right]\phi_i(x). \tag{10}$$

The PDF $\rho(\{\xi_\nu\})$ is a prior on the possible values of the behavioral parameters; we take the prior to be uniform throughout this work. The angle brackets indicate an ensemble average over realizations of the neuronal noise.

We next substitute Eq. (5) into the deviations in Eq. (10) and solve for the connection coefficients κ_ν that minimize the cost function in Eq. (9). The coefficients can then be used to directly calculate the decoding functions using Eq. (5).

A similar procedure is employed to calculate the encoding functions. The target PDF is used to evaluate the firing rates $a_i(t)$ by means of Eq. (4), and the difference of these firing rates from the assumed firing rate profiles $\tilde{a}_i(\{\xi_\nu\})$ serves to determine a quadratic error measure E_{enc}. The coefficients $\hat{\kappa}_{i\nu}$ that minimize E_{enc} are found and utilized to calculate the encoding functions of Eq. (6).

As a demonstration of how a representation can be defined, consider a set of ten firing rate profiles that assume Gaussian form in response to a specific stimulus ξ [Fig. 2(a)]. We take the target PDF to be a Dirac δ function $\delta(x-\xi)$ centered at the stimulus value. We define the minimal space to be the span of the neural activation functions, and generate the encoding functions, decoding functions, and associated transformation coefficients using the procedures described above. The encoding functions have Gaussian forms identical to the corresponding activation functions, while the decoders have a typical structure as shown in Fig. 2(b).

The explicit space representation is reasonably robust against noise. Since the decoded PDF is insensitive to the presence of biologically plausible levels of noise, we will not further consider neuronal noise in this work.

II. NEURAL BELIEF NETWORKS

A. Bayesian belief networks

In Sec. I B, we have summarized methods for encoding and decoding probability density functions into and from the firing rates of populations of neurons. These methods are oriented towards encoding a single random variable (or vector), but we do not wish to restrict ourselves to only the simplest implicit spaces. In this work, we will explore ways in which we can apply the methods so far developed [2] to more complicated implicit spaces. In particular, we will use Bayesian belief networks to efficiently organize the implicit random variables, and then use these Bayesian belief networks to generate neural networks.

Bayesian belief networks are directed acyclic graphs that represent probabilistic models (Fig. 3). Each node represents a random variable, and the arcs (or directed line segments) signify the presence of direct causal influences between the linked variables. The strengths of these influences are defined using conditional probabilities. The directionality of a specified link indicates the direction of causality (or, more simply, relevance); an arc points from direct cause to effect.

Bayesian belief networks have two properties that we will find very useful, both of which stem from the independencies shown by the graph structure. First, the value of a node X is not dependent upon all of the other graph nodes. Rather, it depends only on a subset of the nodes, called a Markov blanket of X, which separates node X from all the other nodes in the graph. The Markov blanket of interest to us can be readily determined from the graph structure. It is comprised of the union of the direct parents of X, the direct successors of X, and all direct parents of the direct successors of X. Second, the joint probability over the random variables is decomposable as

$$P(x_1,x_2,\ldots,x_n)=\prod_{i=1}^{n}P[x_i|N_p(X_i)], \tag{11}$$

where $N_p(X_i)$ refers to the (possibly empty) set of direct parent nodes of X_i. This decomposition comes about from repeated application of Bayes' rule and from the structure of the graph.

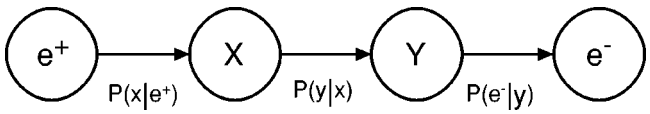

FIG. 3. A chain-structured Bayesian belief network. Evidence e^+ and e^- from the two ends of the chain influences the belief in the random variables X and Y. In a straightforward terminology, X is referred to as the parent of Y and Y as the child of X. From the structure of the graph, we can see, for example, that Y is conditionally independent of e^+ given X; this is true regardless of the values of the links $P(e^-|y)$, $P(y|x)$, and $P(x|e^+)$.

B. Probabilistic inference performed by neural networks

Before exploring arbitrary Bayesian belief networks, it is enlightening to consider a network with a simple graph consisting of two connected nodes $X \to Y$. This graph represents any probabilistic model where a single random variable is inferred from one source of evidence. For convenience, we will work in the minimal space.

Our objective is to find the most suitable marginal PDFs $\rho(y;t)$ and $\rho(x;t)$ to describe the system. We represent the PDFs using Eq. (1) and

$$\rho(y;t) = \sum_\nu B_\nu(t)\Psi_\nu(y). \tag{12}$$

In this network, we find values of the output firing rates $\{B_\nu(t)\}$ only, with input firing rates $\{A_\mu(t)\}$ fully determined by an encoding process.

We define a cost function by

$$\begin{aligned} E_y &= \frac{1}{2}\int \left(\rho(y;t) - \int \rho(y|x)\rho(x;t)dx \right)^2 dy \\ &= \frac{1}{2}\int \left(\sum_\nu B_\nu(t)\Psi_\nu(y) \right. \\ &\quad \left. - \sum_\mu A_\mu(t) \int \rho(y|x)\Phi_\mu(x)dx \right)^2 dy. \end{aligned} \tag{13}$$

The cost function has a minimum corresponding to taking a weighted average of the conditional probability $\rho(y|x)$,

$$\rho(y;t) = \int \rho(y|x)\rho(x;t)dx. \tag{14}$$

We assume in Eq. (14) that the relationship between x and y is independent of the values of the underlying parameters $A_\mu(t)$ of the minimal space [2].

The mean-square error is not the only possibility for the cost function E_y (or the generalized version introduced in Sec. II D). Other appropriate choices may be, e.g., cross entropy or mutual information. Changing the cost function may produce improved results in some cases, and will alter the neural network dynamics we present here.

We minimize E_y using the gradient descent approach, obtaining the update rule

$$\begin{aligned} \frac{dB_\nu(t)}{dt} &= -\eta \frac{\partial E_y}{\partial B_\nu} \\ &= -\eta \left(B_\nu(t) - \sum_\mu \Omega_{\nu\mu} A_\mu(t) \right), \end{aligned} \tag{15}$$

where η is a rate constant and we identify the quantities

$$\Omega_{\nu\mu} = \int \int \Psi_\nu(y)\rho(y|x)\Phi_\mu(x)dxdy. \tag{16}$$

Equation (15) has a fixed point at

$$B_\nu(t) = \sum_\mu A_\mu(t) \int \int \Psi_\nu(y)\rho(y|x)\Phi_\mu(x)dxdy. \tag{17}$$

This is identical to the result obtained by encoding the inference relation in Eq. (14) into the minimal space using Eq. (3).

Equations (15) and (16) define a network in the minimal space which can be converted to a neural network in the explicit space using transformations like those described in Sec. I B. In particular, we make use of Eqs. (7) and (8) and analogous expressions for the random variable Y:

$$B_\nu(t) = \sum_j \kappa^{(y)}_{\nu i} b_i(t), \tag{18}$$

$$b_i(t) = g\left(\sum_\nu \hat{\kappa}^{(y)}_{i\nu} B_\nu(t) \right). \tag{19}$$

Using Eq. (15), the neural firing rates, for small times τ, become

$$b_i(t+\tau) = g\left(\sum_j \lambda_{ij} b_j(t) + \sum_i \omega_{ij} a_i(t) \right), \tag{20}$$

where

$$\lambda_{ij} = (1-\eta\tau) \sum_\nu \hat{\kappa}^{(y)}_{i\nu} \kappa^{(y)}_{\nu j}, \tag{21}$$

$$\omega_{ij} = \eta\tau \sum_{\mu,\nu} \hat{\kappa}^{(y)}_{i\nu} \Omega_{\nu\mu} \kappa_{\mu j}. \tag{22}$$

The neural network features lateral weights λ_{ij} that act as a form of associative memory, driving the preservation of a fixed neural state, and feedforward weights ω_{ij} that implement probabilistic inference, driving the neural activation state towards the fixed point given in Eq. (17). The quantity $\eta\tau$ determines the relative importance of the two effects, allowing us to tune the network dynamics for different systems.

The neural network in Eq. (20) is defined for discrete time steps τ. It is also possible to generate neural networks evolving in continuous time. The network dynamics can be determined by applying the chain rule to Eq. (19) to find db_i/dt and eliminating all of the minimal space variables in favor of the explicit space neural firing rates. Alternatively, a useful approximation can be obtained by expanding $b_i(t+\tau)$ in Eq. (20) to first order in τ, giving rise to

$$\tau \frac{db_i(t)}{dt} = -b_i(t) + g\left(\sum_j \lambda_{ij} b_j(t) + \sum_i \omega_{ij} a_i(t) \right). \tag{23}$$

C. Predictive and retrospective support

Neural belief networks can feature two distinct types of information propagation that provide support for the PDFs represented at each graph node. Predictive support, also called causal support, is probabilistic information that propa-

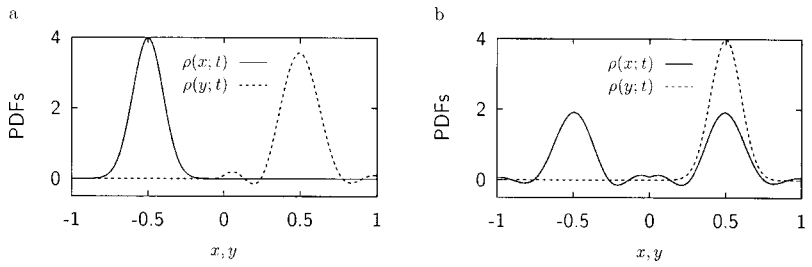

FIG. 4. Predictive and retrospective support of the absolute-value function. The results shown here utilize spaces of dimension $D=6$ to represent both x and y in the two networks. The structure of the spaces was determined from the singular-value decomposition [13] of a discrete approximation of $\rho(y|x)$—the basis functions $\Phi_\mu(x)$ and $\Psi_\nu(y)$ are set to the singular vectors corresponding to the D largest singular values. (a) The network with predictive support closely approximates the absolute-value function. Specifying a PDF $\rho(x;t)$ centered about $x=-1/2$ yields an inferred PDF $\rho(y;t)$ of similar form centered about $x=+1/2$. (b) The network with retrospective support allows for both possible solutions. Specifying a unimodal PDF $\rho(y;t)$ centered about $x=+1/2$ yields an inferred bimodal PDF $\rho(x;t)$, with the modes centered about $x=-1/2$ and $x=+1/2$.

gates, along the directions of the graph links, from cause to effect [4]. The network considered in Sec. II B involves only predictive support.

The second type of support is retrospective, or diagnostic, support. In this case, information propagates against the directions of the graph links, from effect to cause, or, equivalently, from evidence to hypothesis [4]. By specifying $\rho(y;t)$ instead of $\rho(x;t)$, the $X\rightarrow Y$ inference network features only retrospective support.

Using the same minimal-space representation [Eqs. (1) and (12)] as we adopted for the predictive network, we determine the update rule for the retrospective network. We find

$$\frac{dA_\mu(t)}{dt}=\eta\left(\sum_\beta B_\beta(t)\Omega_{\beta\mu}-\sum_\alpha A_\alpha(t)\Upsilon_{\alpha\mu}\right) \tag{24}$$

with feedback weights

$$\Omega_{\beta\mu}=\int\int \Psi_\beta(y)\rho(y|x)\Phi_\mu(x)dxdy \tag{25}$$

and lateral weights

$$\Upsilon_{\alpha\mu}=\int\left(\int \rho(y|x)\Phi_\alpha(x)dx\right)\left(\int \rho(y|x)\Phi_\mu(x)dx\right)dy. \tag{26}$$

While the feedback weights $\Omega_{\beta\mu}$ of the retrospective network are identical to the feedforward weights of the predictive network [Eq. (15)] in the minimal space, note that the resulting neural-network weights in the explicit space need not be identical. The lateral weights $\Upsilon_{\alpha\mu}$ provide a measure of the correlation between what the different basis functions in the parent node X predict about the child node Y; these lateral connections act to ensure consistency between evidence and hypothesis.

Although the networks driven by retrospective support are closely related to the networks driven by predictive support, their function is quite different. For example, consider a network with $\rho(y|x)=\delta(y-|x|)$ as the underlying computation. In a predictive network, wherein we specify $\rho(x;t)$ and infer $\rho(y;t)$, the absolute-value relationship is approximated in a straightforward fashion [Fig. 4(a)]. Conversely, a retrospective network based upon the same conditional PDF is called upon to ''invert'' the absolute value, a noninvertible function. The inferred $\rho(x;t)$ decoded from the retrospective network [Fig. 4(b)] captures both of the possible solutions—with positive and negative values—in response to a unimodal PDF $\rho(y;t)$.

D. Encoding Bayesian belief networks into neural networks

Following a strategy similar to that presented in Sec. II B, we can develop neural-network update rules from arbitrary Bayesian belief networks . We assume that the Bayesian belief network consists of R nodes, symbolizing random variables $X_1,X_2,X_3,\ldots,X_R$. We focus on the marginal distributions $\rho(x_i;t)$ for the random variables as a function of time. The overall joint probability density is calculated as the product of the marginal distributions, i.e., a so-called naive estimate.

Introducing representations

$$\rho(x_i;t)=\sum_\mu A_\mu^{(i)}(t)\Phi_\mu^{(i)}(x_i) \tag{27}$$

for the marginal distributions in the minimal space, we define auxiliary cost functions, analogous to that in Eq. (13), with the forms

$$\begin{aligned}E_i&=\frac{1}{2}\int\left(\rho(x_i;t)-\int\rho[x_i|N_p(X_i)]\prod_j\rho(x_j;t)dx_j\right)^2dx_i\\&=\frac{1}{2}\int\left[\sum_\mu A_\mu^{(i)}(t)\Phi_\mu^{(i)}(x_i)\right.\\&\quad\left.-\int\rho[x_i|N_p(X_i)]\prod_j\sum_{\nu_j}A_{\nu_j}^{(j)}(t)\Phi_{\nu_j}^{(j)}(x_j)dx_j\right]^2dx_i.\end{aligned} \tag{28}$$

In Eq. (28), the index j for the product runs over the direct parents of X_i.

We further define an aggregate cost function

$$E=\sum_{i=1}^R K_iE_i. \tag{29}$$

The parameters K_i can be used to emphasize particular portions of the network; except where otherwise noted, we assume that $K_i=1$ for all i. Employing gradient descent to minimize E, we find

$$\frac{dA_\sigma^{(k)}}{dt} = -\eta \sum_{i=1}^{R} K_i \frac{\partial E_i}{\partial A_\sigma^{(k)}}. \tag{30}$$

Since $A_\sigma^{(k)}$ does not appear in all of the cost functions, $\partial E_i / \partial A_\sigma^{(k)}$ is nonzero only for $i=k$ and when the graph node for X_i is $N_c(X_k)$, the set of children of the graph node for X_k. Thus,

$$\frac{1}{\eta}\frac{dA_\sigma^{(k)}}{dt} = -K_k \frac{\partial E_k}{\partial A_\sigma^{(k)}} - \sum_{i \in \{i: X_i \in N_c(X_k)\}} K_i \frac{\partial E_i}{\partial A_\sigma^{(k)}}. \tag{31}$$

As was the case for the two-node inference network, the input PDF or PDFs will be specified by an encoding process rather than an update rule of this sort.

The derivatives in Eq. (31) are straightforward but lengthy to evaluate. The resulting general equations (presented in the Appendix) are lengthy and can sometimes be awkward to use directly. For specific probabilistic models, it may be more convenient to write out the cost functions using Eq. (28) and directly evaluate the necessary derivatives [specified by Eq. (31)].

III. APPLICATIONS OF NEURAL BELIEF NETWORKS

A. Bidirectional propagation

So far, we have restricted our attention to developing neural networks that encode simple probabilistic models involving only a single source of evidence. Given suitable representations, we can design neural networks that capture a wide variety of probabilistic relations [2]. Further, by regarding representations of probabilistic dependence models as Bayesian belief networks, we have seen that two types of information propagation come into play: predictive and retrospective.

In this section, we will examine several applications of neural belief networks. Unlike those considered before, the probabilistic models will now feature multiple sources of evidence, with bidirectional propagation of both predictive and retrospective support. The corresponding neural networks thus have neurons with both feedforward and feedback connections, as well as lateral connections.

B. Neural propagation of evidence in trees

To facilitate investigation of information propagation in neural belief networks, we first focus on networks whose implicit spaces are specified by tree-structured Bayesian belief networks [Fig. 5(a)]. These implicit networks are general enough to illustrate the concepts, while yielding neural networks that are readily understood. We will examine binary trees, where each node has at most two children, but the results extend simply to more general tree structures.

We may assume that evidence is only available in the root node and the leaf nodes (although all such nodes need not provide evidence). The root node is the single node which has no father and is located at the top of the tree, while the leaf nodes are all the nodes which have no children. If another node X were externally specified, the subtree rooted at X could be broken off and treated separately. Conversely, the father node of X is unaffected by the descendants of X, so they could be deleted from the original tree, leaving X as a leaf node.

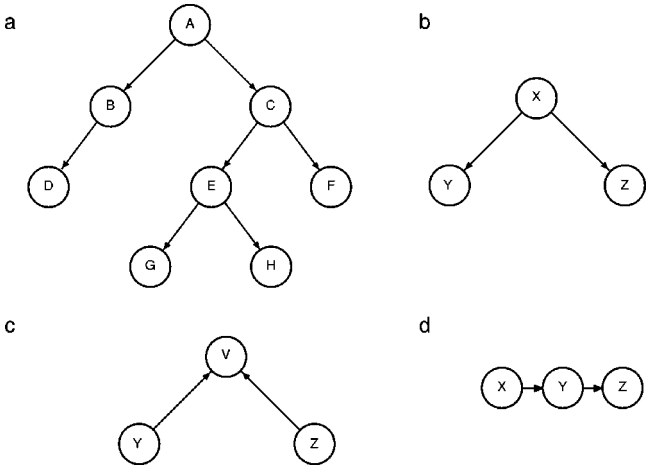

FIG. 5. Tree-structured Bayesian belief networks. (a) In this tree, node A is called the root, while nodes D, F, G, and H are called the leaves. For trees, the direct parent of a node is called its father, and its direct children are called its sons. Since each father has at most two sons, the tree shown here is a binary tree. (b) A small tree. Any of the nodes in this tree can be specified as evidence. The two leaf nodes could also provide evidence: both leaves can provide information to the root, but if the root and one of the leaves is specified, the other leaf will only be driven by the root (its Markov blanket). (c) A treelike graph with the arrows reversed. Any two of the nodes can be specified and its information will propagate throughout the network. Unlike the case of the tree, the Markov blanket of any node is both of the other nodes. (d) Chains are special cases of trees. This three-node chain can feature both predictive and retrospective support.

Clearly, an unspecified root node can only receive retrospective support, while an unspecified leaf node can only receive predictive support. All other unspecified nodes in the tree will receive both retrospective and predictive support. We will thus need to consider separately these three types of nodes when determining the update rules for the neural network.

An unspecified root node X with children Y and Z receives feedback inputs (retrospective support) from both of them (Fig. 5). We introduce the representations

$$\rho(x;t) = \sum_\alpha A_\alpha(t)\Phi_\alpha(x), \tag{32}$$

$$\rho(y;t) = \sum_\beta B_\beta(t)\Psi_\beta(y), \tag{33}$$

$$\rho(z;t) = \sum_\gamma C_\gamma(t)\Theta_\gamma(z). \tag{34}$$

Following the procedures described in Sec. II D, the update rule for the root is

$$\frac{1}{\eta}\frac{dA_\mu}{dt} = -K_y\frac{\partial E_y}{\partial A_\mu} - K_z\frac{\partial E_z}{\partial A_\mu}, \tag{35}$$

with

$$\begin{aligned}\frac{\partial E_y}{\partial A_\mu} = &\sum_\alpha A_\alpha(t)\int\left(\int \rho(y|x)\Phi_\alpha(x)dx\right)\\ &\times\left(\int \rho(y|x)\Phi_\mu(x)dx\right)dy\\ &-\sum_\beta B_\beta(t)\int\int \Psi_\beta(y)\rho(y|x)\Phi_\mu(x)dxdy\end{aligned} \tag{36}$$

and

$$\begin{aligned}\frac{\partial E_z}{\partial A_\mu} = &\sum_\alpha A_\alpha(t)\int\left(\int \rho(z|x)\Phi_\alpha(x)dx\right)\\ &\times\left(\int \rho(z|x)\Phi_\mu(x)dx\right)dz\\ &-\sum_\gamma C_\gamma(t)\int\int \Theta_\gamma(z)\rho(z|x)\Phi_\mu(x)dxdz.\end{aligned} \tag{37}$$

Thus, the firing rates for the root node are driven by a sum of feedback inputs that individually are identical to the input produced by a single source of retrospective support (Sec. II C). The parameters K_y and K_z need not be the same; different values may be used to give greater significance to one of the inputs.

The firing rates for an unspecified leaf node are also updated by a familiar rule. Consider a leaf node X with father U. Using Eq. (32) and

$$\rho(u;t) = \sum_\delta D_\delta(t)\Lambda_\delta(u), \tag{38}$$

we obtain the update rule

$$\frac{1}{\eta}\frac{dA_\mu}{dt} = -K_x\frac{\partial E_x}{\partial A_\mu}, \tag{39}$$

where

$$\frac{\partial E_x}{\partial A_\mu} = A_\mu(t) - \sum_\delta D_\delta(t)\int\int \Lambda_\delta(u)\rho(x|u)\Phi_\mu(x)dudx. \tag{40}$$

This update rule is of course identical to the update rule for the $X\rightarrow Y$ network with predictive support (Sec. II B).

All nonroot, nonleaf nodes have similar update rules. For a node X with father U and sons Y and Z, we impose the representations given above in Eqs. (32)–(34) and (38). Applying the procedure of Sec. II D yields an update rule with form

$$\frac{1}{\eta}\frac{dA_\mu}{dt} = -K_y\frac{\partial E_y}{\partial A_\mu} - K_z\frac{\partial E_z}{\partial A_\mu} - K_x\frac{\partial E_x}{\partial A_\mu}. \tag{41}$$

The partial derivatives of the cost functions are identical to those evaluated for the root and leaf nodes. The descendants and ancestors of X thus communicate, in the neural network, only through the intermediary of X itself. This is consistent with the tree structure of the underlying Bayesian belief network; given X, the descendants of X are conditionally independent of the ancestors of X [4].

With these update rules, evidence provided at the root node or at leaf nodes will propagate throughout the network. The manner in which evidence is specified will depend on how the probabilistic model is posed. Therefore, the same graph could have different nodes specified for different purposes. For instance, the small tree in Fig. 5(b) could have any of its nodes represent sensory inputs.

If we specify the PDF for the root node, the leaf nodes Y and Z will receive predictive support from X. By selecting appropriate representations for the PDFs and for the conditional probabilities associated with the links, the resulting neural network could subdivide a complex, highly general sensory input into simpler, more specialized components. For example, a visual input at a particular retinal location might be separated into contrast and color.

Conversely, if we specify PDFs for the leaf nodes, the root node X will receive retrospective support from Y and Z. Amongst other possibilities, this provides a simple way to model redundant sensory inputs. If we take the conditional probabilities to be of narrow Gaussian form, $\rho(y|x) = N(y;x,\sigma_y^2)$ and $\rho(z|x) = N(z;x,\sigma_z^2)$, then the firing rates representing $\rho(x;t)$ will be updated so as to pool the diagnostic information from both the sensory inputs, Y and Z.

Similar arguments apply to larger trees. Additionally, larger trees may well have the root node and leaf nodes specified simultaneously (which is not of interest for the small tree discussed above). These nodes may correspond to sensory inputs, or can represent priors that are built into the neural network.

It is important to recognize that the neural update rules developed above apply only to the binary trees. If the arrows in the binary tree graphs are reversed, rather different update rules are produced. For example, the directions of the small tree shown in Fig. 5(b) can be reversed, as shown in Fig. 5(c). Specifying Y and Z yields the multiplicative update rule

$$\begin{aligned}&\frac{1}{\eta K_x}\frac{dA_\alpha(t)}{dt}\\ &\quad = -A_\alpha(t) + \sum_{\beta,\gamma} B_\beta(t)C_\gamma(t)\\ &\qquad\times\int\int\int \Phi_\alpha(x)\rho(x|y,z)\Psi_\beta(y)\Theta_\gamma(z)dxdydz,\end{aligned} \tag{42}$$

while specifying X and Z yields

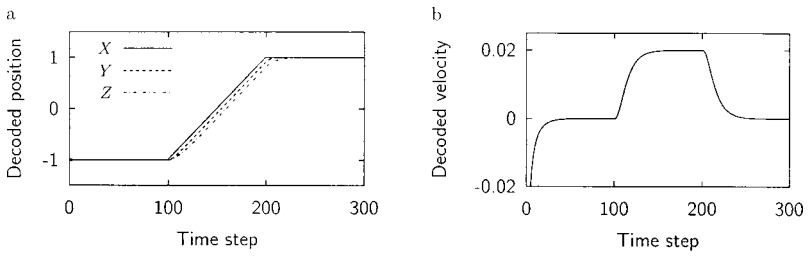

FIG. 6. A neural belief network can estimate the velocity of a moving target. (a) The position of the target is copied into two different populations of neurons, with different time delays. (b) The time delay and the difference of the two copies of position are used to estimate the velocity. The results shown here were obtained with the PDFs for each of the random variables represented using minimal spaces spanned by two straight-line basis functions.

$$\frac{1}{\eta K_x}\frac{dB_\nu(t)}{dt} = \sum_{\alpha,\gamma} A_\alpha(t)C_\gamma(t)\int\int\int \Phi_\alpha(x)\rho(x|y,z)\Psi_\nu(y) \times \Theta_\gamma(z)dxdydz - \sum_{\beta,\gamma_1,\gamma_2} B_\beta(t)C_{\gamma_1}(t)C_{\gamma_2}(t) \times \int\left\{\left(\int\int \rho(x|y,z)\Psi_\beta(y)\Theta_{\gamma_1}(z)dydz\right) \times\left(\int\int \rho(x|y,z)\Psi_\nu(y)\Theta_{\gamma_2}(z)dydz\right)\right\}dx. \quad (43)$$

This latter update rule features a feedforward term that is multilinear in the parameters $\{A_\alpha(t)\}$ and $\{C_\gamma(t)\}$. It also features a nonlinear lateral combination of the parameters $\{B_\beta(t)\}$, and $\{C_\gamma(t)\}$ which serves to ensure that the two parent nodes Y and Z are mutually consistent with the PDF of the child node X. If only X is specified, there will be an additional update rule for the parameters $\{C_\gamma(t)\}$, which is similar in form to Eq. (43).

Although the update rules are more complicated with the directions reversed in this manner, the probabilistic model may demand it. For instance, the Bayesian belief network in Fig. 5(c) is appropriate for implementing the arithmetic operations (add, subtract, multiply, and divide).

An interesting application of these two types of neural belief networks (tree and reversed tree) is the estimation of the velocity of a moving object. A small tree [Fig. 5(b)] can generate two copies Y and Z of the input position X by taking the conditional probabilities to be Dirac δ functions $\delta(y-x)$ and $\delta(z-x)$. We can set the parameters K_i so that the values of the copies will be held for different lengths of time. In particular, we can establish the relations $\rho(y;t)=\rho(x;t-\tau)$ and $\rho(z;t)=\rho(x;t-2\tau)$. These copies of the position can then be used as the inputs to a second Bayesian belief network , of the type shown in Fig. 5(c). By setting $\rho(v|y,z)=\delta[v-(y-z)/\tau]$ in this second network, the random variable V becomes an estimate of the velocity at time $t-\tau$.

However, there is not a unique way to extract the velocity from the network. We have chosen to obtain the velocity as the mean value of the random variable V (Fig. 6). This choice presents a difficulty for ambiguous cases, e.g., a multimodal PDF $\rho(x;t)$, where the mean can be a poor match to any of the possible velocities. One approach to eliminating problems caused by ambiguous inputs is to utilize a low-dimensional representation where the mean is well represented but multimodal distributions are excluded; the velocity estimates in Fig. 6 were produced using such a representation (discussed at length in Ref. [2]). Alternatively, the velocity could be determined in a different fashion [e.g., taking the maximum value of $\rho(v;t)$], or the network could be implemented to directly handle ambiguous cases (see the following section).

C. Top-down feedback from a high-level model

In Sec. III B, we examined the neural update rules for Bayesian belief networks having binary-tree structure. By doing so, we examined the means by which information propagates throughout the network from one or more sources. In particular, we saw that conditional independence in the Bayesian belief network was preserved in the neural-network connectivity.

We now turn from the general rules by which probabilistic information propagates in the neural network and investigate in more detail the effects that multiple sources of evidence have on the encoded PDFs. We consider PDFs encoded by chains of nodes with evidence provided at one or both ends. (Specifying the PDF for any other node breaks the chain into two chains that can be treated independently.) A chain is a special case of a tree, so the neural update rules can be obtained as in Sec. III B.

We have already studied at some depth the behavior of chains consisting of two nodes (Secs. II B and II C). These chains are able to transmit probabilistic information from one node of the graph to another, with the accuracy limited by the representations adopted. Even a relatively small number of neurons can produce networks of suitable quality (e.g., the networks implementing the absolute value function in Fig. 4).

Every chain with three or more nodes will admit both predictive and retrospective support. The behavior of all such chains is well characterized by a chain with three nodes: the update rule for any node depends only on the neighboring nodes, so a chain with three nodes [Fig. 5(d)] covers all

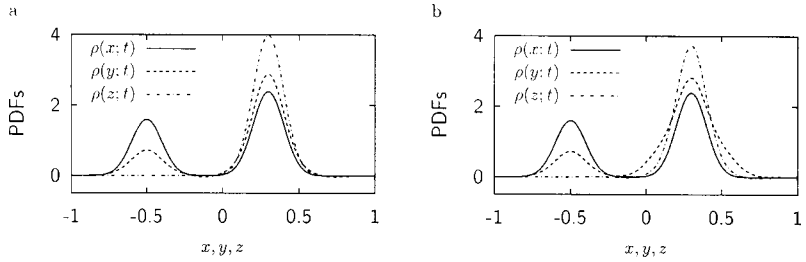

FIG. 7. Propagation of evidence in chain-structured neural belief networks. (a) Multiple sources of evidence can help to resolve ambiguous information. Here, the inferior mode of a bimodal, bottom-up input $\rho(x;t)$ is damped by a more specific top-down signal $\rho(z;t)$. (b) A high-level model can be dynamically generated in a neural belief network with a population Z of winner-take-all neurons. An ambiguous input signal at X is propagated to the winner-take-all neurons through Y. The winner-take-all neurons only respond to the stronger mode of the bimodal input, which damps the inferior mode in $\rho(y;t)$.

possibilities (only predictive support, only retrospective support, and both predictive and retrospective support). To keep the focus on the interaction of multiple sources of evidence, we take the conditional probabilities to be the Dirac δ functions $\rho(y|x)=\delta(y-x)$ and $\rho(z|y)=\delta(z-y)$ and use identical parameters K_y and K_z. We utilize the Gaussian representation introduced in Sec. I C

The first possibility is just to add the third node without adding any additional evidence to the network. We decode $\rho(y;t)$ and $\rho(z;t)$ from neural firing rates determined using update rules derived previously for more general trees. For identical inputs in the two-node and three-node networks, the steady state behavior of $\rho(y;t)$ is unchanged by the addition of the retrospective support from the third node, and the structure of $\rho(z;t)$ is identical to that of $\rho(y;t)$.

Since the same evidence is presented to the network, the introduction of a third node does not change the behavior of the original nodes. This is entirely appropriate, given the probabilistic foundations of the neural belief networks. However, it is feasible that, by adjusting the K_i parameters or the PDFs representable by the minimal spaces, there can be a change in the dynamics of neural networks extended in this fashion. For example, appropriate parameter choice could produce a slowly varying PDF in Z, which is relatively stable against noise, thus stabilizing a more rapidly varying PDF in Y.

A more proactive role that the third node Z can play is as an additional source of evidence. We directly specify $\rho(z;t)$ and encode it into the neural network. The neural firing rates for Y are driven by predictive support from X and retrospective support from Z, and then decoded to find $\rho(y;t)$. One possible use of this second source of evidence is to resolve an ambiguous input: the inferior mode of a bimodal predictive input can be deemphasized using more specific retrospective evidence [Fig. 7(a)].

Although this use of the retrospective evidence resolves the ambiguous predictive input, it does not explain how the retrospective evidence comes about. Ideally, we would like to build up a high-level model of the predictive input, and use the model to generate a top-down signal that imposes global regularity on the bottom-up predictive input. This requires an extension of the probabilistic framework to allow one to assess the quality of a PDF as a model, rather than simply to employ PDFs as descriptions of analog variables.

To do this, we adopt an approach that illustrates the first steps towards implementing decision theory [14] in neural networks. Conceptually, we introduce a set of model PDFs, and require $\rho(z;t)$ to be the PDF from this set that is the closest match to $\rho(y;t)$. For the present example, we take the model PDFs to have Gaussian forms, similar to those in Fig. 2(a), but suitably normalized.

The model PDFs are translated into the explicit-space neural network by taking the neurons representing $\rho(z;t)$ to be winner-take-all units. Only one of these neurons will be active at a time, based on a simple utility function: the neurons compete to be active, and the single neuron that is most strongly driven will be the one that is activated. The manner of implementation of the winner-take-all units is immaterial, so we directly choose the most strongly driven neuron in the computer simulations. (It is possible to implement a set of winner-take-all units in a real network through a suitable choice of nonlinear activation function and lateral weights. One may, e.g., let each neuron inhibit the others and have a self-excitatory connection [15].)

The decoding functions used earlier are no longer well suited for Z. They were optimized to collectively represent general PDFs, but individually they work poorly to represent the model PDFs. However, it is clear that, for the winner-take-all neurons we have introduced, the appropriate decoders are directly proportional to the model PDFs themselves. This leads to minor alterations in the weights of the neural network in the explicit space.

Since only the most strongly driven winner-take-all neuron is activated, this strategy provides us with a way to generate a high-level model that selects the dominant mode of a multimodal input distribution. We allow Z to be driven by the predictive support from Y, and the winner-take-all nature of the Z neurons permits only narrow Gaussian PDFs to be represented. Thus, $\rho(z;t)$ serves as our high-level model, and can resolve ambiguous inputs, as demonstrated in Fig. 7(b). This type of neural network could be used as the starting point for coherent theoretical accounts of attentive effects in the primate visual system, of the electrosensory system of weakly electric fish, and of other neural systems where an internal model is built up to impose global constraints on neural representations of information.

IV. CONCLUSIONS

We have extended the hypothesis that neural networks represent information as probability density functions. These PDFs are assumed to be obtainable by a linear combination of some implicit decoding functions, with the decoder for each neuron being weighted by its firing rate. The firing rates in turn are obtainable from the PDF using a complementary set of encoding functions.

Success in representing an individual PDF with a population of neurons [2] has led us to inquire whether more complex probabilistic models can be represented and implemented in neural terms. To this end, we have adopted graphical representations of probabilistic dependence models, in the form of Pearl's Bayesian belief networks, to organize and simplify the relations between the random variables in such a model.

The resulting neural belief networks share properties with both neural-network models and Bayesian belief networks. In particular, multiple sources of evidence are consistently pooled based on local update rules, providing a distributed version of a probabilistic model. A number of interesting applications of neural belief networks are possible, including velocity estimation and the disambiguation of low-level inputs using feedback from a high-level model.

A possible shortcoming of the class of networks explored here is that the dynamics depend upon multiplicative responses [Eqs. (42), (43) and (A1)]. Multiplication of the minimal-space coefficients becomes multiplication of neural firing rates in the explicit space. Neurons with multiplicative responses are computationally powerful, but single-neuron mechanisms for neuronal multiplication have proven elusive. Salinas and Abbott have shown that populations of neurons can collectively produce multiplicative effects [16], but it is not clear that the precise and complex multiplication needed can be implemented in this way. It may be necessary to generalize the simple encoding rule [Eq. (4)] to a more sophisticated form that produces neural networks with less dependence on multiplicative interactions.

In summary, we have introduced, analyzed, and applied three ways to represent probabilistic information. These are the implicit model, depicted as a Bayesian belief network; the representation of the probabilistic model in the minimal space; and the representation of the probabilistic model as a neural network in the explicit space. Further, we have devised rules that permit us to convert one type of representation into another type. This formalization of the representation and processing of information in neurobiological computation therefore suggests a general protocol by which probabilistic models can be embedded in neural networks. First, we specify the probabilistic model, using a Bayesian belief network to organize the random variables. We then consider what types of functions are exemplars of the PDFs describing the random variables and use these exemplars to define the minimal space. Finally, we utilize the relations between the minimal space and the explicit space to generate the neural network itself.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation (Grant Nos. IBN-9634314, PHY-9602127, PHY-9900713, and PHY-0140316), the DFG of Germany (Graduiertenkolleg Azentrische Kristalle, GK549), and the FCT of Portugal (Bolsa de investigação do CCM and SFRH/BPD/9417/2002). Portions of this work were undertaken while M.J.B. and J.W.C. were participants in the Research Year on ''The Sciences of Complexity: From Mathematics to Technology to a Sustainable World'' at the Zentrum für interdisziplinäre Forschung, University of Bielefeld. J.W.C. also acknowledges support received from the Fundação Luso-Americana para o Desenvolvimento (FLAD) and from the Fundação para a Ciência e a Technologia (FCT) for his participation in Madeira Math Encounters XXIII at the University of Madeira, where the work was concluded.

APPENDIX: GENERAL EQUATIONS FOR NEURAL BELIEF NETWORKS

$$\frac{1}{\eta}\frac{dA_\sigma^{(k)}}{dt} = -K_k\left(A_\sigma^{(k)} - \prod_j \sum_{\nu_j} A_{\nu_j}^{(j)} \Omega^{(k)}_{\sigma\nu_1\nu_2\cdots\nu_{j_{max}}}\right) + \sum_i K_i \sum_\mu A_\mu^{(i)} \prod_{j\neq k} \sum_{\nu_j} A_{\nu_j}^{(j)} \Omega^{(ik)}_{\mu\nu_1\nu_2\cdots\nu_{j_{max}}\sigma} - \sum_i K_i \prod_j \prod_{j'\neq k} \sum_{\nu_j} A_{\nu_j}^{(j)} \sum_{\mu_{j'}} A_{\mu_{j'}}^{(j')} \times \Upsilon^{(jj'k)}_{\nu_1\nu_2\cdots\nu_{j_{max}}\mu_1\mu_2\cdots\mu_{j'_{max}}\sigma}, \tag{A1}$$

where we have defined

$$\Omega^{(k)}_{\sigma\nu_1\nu_2\cdots\nu_{j_{max}}} = \int \Phi_\sigma^{(k)}(x_k)\rho[x_k|N_p(x_k)]\prod_j \Phi_{\nu_j}^{(j)}(x_j)dx_j dx_k \tag{A2}$$

$$\Omega^{(ik)}_{\mu\nu_1\nu_2\cdots\nu_{j_{max}}\sigma} = \int \Phi_\mu^{(i)}(x_i)\rho[x_i|N_p(x_i)]\Phi_\sigma^{(k)}(x_k) \times \prod_{j\neq k} \Phi_{\nu_j}^{(j)}(x_j)dx_j dx_i dx_k, \tag{A3}$$

and

$$\Upsilon^{(jj'k)}_{\nu_1\nu_2\cdots\nu_{j_{max}}\mu_1\mu_2\cdots\mu_{j'_{max}}\sigma} = \int \left(\int \Phi_\sigma^{(k)}(x_k)\rho[x_i|N-p(x_i)] \times \prod_{j'\neq k} \Phi_{\mu_{j'}}^{(j')}(x_{j'})dx_{j'}dx_k \int \rho[x_i|N_p(x_i)] \times \prod_j \Phi_{\nu_j}^{(j)}(x_j)dx_j \right) dx_i. \tag{A4}$$

The sums over the index i in Eq. (A1) run over the children of X_k. The products are over the parents of either node X_k

[Eq. (A2), first term in Eq. (A1)] or node X_i [Eqs. (A3) and (A4), second and third terms in Eq. (A1)], possibly excluding node X_k itself. From these equations, it can be seen that the PDF represented at a node X_i is updated based only on its direct parents, its direct descendents, and the direct parents of its direct descendents, so that X_i is separated from all other nodes in the neural belief networks by an appropriate Markov blanket.

[1] C.H. Anderson, Int. J. Mod. Phys. C **5**, 135 (1994).

[2] M.J. Barber, J.W. Clark, and C.H. Anderson, Neural Comput. **15**, 1843 (2003).

[3] D. Van Essen, C.H. Anderson, and D.J. Felleman, Science **255**, 419 (1992).

[4] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, San Mateo, CA, 1988).

[5] P. Smyth, D. Heckerman, and M.I. Jordan, Neural Comput. **9**, 227 (1997).

[6] R.M. Neal, Artif. Intell. **56**, 71 (1992).

[7] R.S. Zemel, in *Theories of Cortical Processing*, edited by R. Hecht-Neilsen (Springer-Verlag, Berlin, 1999).

[8] C.H. Anderson, in *Condensed Matter Theories*, edited by E.V. Ludeñ, P. Vashishta, and R.F. Bishop (Nova Science, Commack, NY, 1996), Vol. 11, pp. 365–374.

[9] A.P. Georgopoulos, A.B. Schwartz, and R.E. Kettner, Science **233**, 1416 (1986).

[10] A.B. Schwartz, J. Neurophysiol. **70**, 28 (1993).

[11] F. Rieke, D. Warland, R.R. de Ruyter van Steveninck, and W. Bialek, *Spikes: Exploring the Neural Code* (MIT Press, Cambridge, MA, 1997).

[12] R.S. Zemel, P. Dayan, and A. Pouget, Neural Comput. **10**, 403 (1998).

[13] J.L. Goldberg, *Matrix Theory With Applications* (McGraw-Hill, New York, 1991).

[14] H.A. Raiffa, *Decision Analysis: Introductory Lectures on Choices Under Uncertainty* (Addison-Wesley, Reading, MA, 1968).

[15] J. Hertz, A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation* (Addison-Wesley, Reading, MA, 1991).

[16] E. Salinas and L.F. Abbott, Proc. Natl. Acad. Sci. U.S.A. **93**, 11956 (1996).