Inference by Reparameterization in Neural Population Codes

Anonymous Author(s)
Affiliation
Address
email

Abstract

Behavioral experiments on humans and animals suggest that the brain performs probabilistic inference to interpret its environment. Here we present a new general-purpose, biologically-plausible neural implementation of approximate inference. The neural network represents uncertainty using Probabilistic Population Codes (PPCs), which are distributed neural representations that naturally encode probability distributions, and support marginalization and evidence integration in a biologically-plausible manner. By connecting multiple PPCs together as a probabilistic graphical model, we represent multivariate probability distributions. Approximate inference in graphical models can be accomplished by message-passing algorithms that disseminate local information throughout the graph. An attractive and often accurate example of such an algorithm is Loopy Belief Propagation (LBP), which uses local marginalization and evidence integration operations to perform approximate inference efficiently even for complex models. Unfortunately, a subtle feature of LBP renders it neurally implausible. However, LBP can be elegantly reformulated as a sequence of Tree-based Reparameterizations (TRP) of the graphical model. We re-express the TRP updates as a nonlinear dynamical system with both fast and slow timescales, and show that this produces a neurally plausible solution. By combining all of these ideas, we show that a network of PPCs can represent multivariate probability distributions and implement the TRP updates to perform probabilistic inference. Simulations with Gaussian graphical models demonstrate that the neural network inference quality is comparable to the direct evaluation of LBP and robust to noise, and thus provides a promising mechanism for general probabilistic inference in the population codes of the brain.

1 Introduction

In everyday life we constantly face tasks we must perform in the presence of sensory uncertainty. A natural and efficient strategy is then to use probabilistic computation. Behavioral experiments have established that humans and animals do in fact use probabilistic rules in sensory, motor and cognitive domains [1, 2, 3]. However, the implementation of such computations at the level of neural circuits is not well understood.

In this work, we ask how distributed neural computations can consolidate incoming sensory information and reformat it so it is accessible for many tasks. More precisely, how can the brain simultaneously infer marginal probabilities in a probabilistic model of the world? Previous efforts to model marginalization in neural networks using distributed codes invoked limiting assumptions, either treating only a small number of variables [4], allowing only binary variables [5, 6, 7], or restricting interactions [8, 9]. Real-life tasks are more complicated and involve a large number of variables that need to be marginalized out, requiring a more general inference architecture.

Submitted to 29th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

Here we present a distributed, nonlinear, recurrent network of neurons that performs inference about many interacting variables. There are two crucial parts to this model: the representation and the inference algorithm. We assume that brains represent probabilities over individual variables using Probabilistic Population Codes (PPCs) [10], which were derived from using Bayes' Rule on experimentally measured neural responses to sensory stimuli. Here for the first time we link multiple PPCs together to construct a large-scale graphical model. For the inference algorithm, many researchers have considered Loopy Belief Propagation (LBP) to be a simple and efficient candidate algorithm for the brain [11, 12, 13, 14, 8, 5, 7, 6]. However, we will discuss one particular feature of LBP that makes it neurally implausible. Instead, we propose that an alternative formulation of LBP known as Tree-based Reparameterization (TRP) [15], with some modifications for continuous-time operation at two timescales, is well-suited for neural implementation in population codes.

We describe this network mathematically below, but the main conceptual ideas are fairly straightforward: multiplexed patterns of activity encode statistical information about subsets of variables, and neural interactions disseminate these statistics to all other encoded variables for which they are relevant.

In Section 2 we review key properties of our model of how neurons can represent probabilistic information through Probabilistic Population Codes. Section 3 reviews graphical models, Loopy Belief Propagation, and Tree-based Reparameterization. In Section 4, we merge these ingredients to model how populations of neurons can represent and perform inference on large multivariate distributions. Section 5 describes experiments to test the performance of network. We summarize and discuss our results in Section 6.

2 Probabilistic Population Codes

Neural responses $\mathbf{r}$ vary from trial to trial, even to repeated presentations of the same stimulus x. This variability can be expressed as the likelihood function $p(\mathbf{r}|x)$. Experimental data from several brain areas responding to simple stimuli suggests that this variability often belongs to the exponential family of distributions with linear sufficient statistics [10, 16, 17, 4, 18]:

$$p(\mathbf{r}|x) = \phi(\mathbf{r}) \exp(\mathbf{h}(x) \cdot \mathbf{r}), \tag{1}$$

where $\mathbf{h}(x)$ depends on the stimulus-dependent mean and fluctuations of the neuronal response and $\phi(\mathbf{r})$ is independent of the stimulus. For a conjugate prior $p(x)$, the posterior distribution will also have this general form, $p(x|\mathbf{r}) \propto \exp(\mathbf{h}(x) \cdot \mathbf{r})$. This neural code is known as a linear PPC: it is a Probabilistic Population Code because the population activity collectively encodes the stimulus probability, and it is linear because the log-likelihood is linear in $\mathbf{r}$. In this paper, we assume responses are drawn from this family, although incorporation of more general PPCs with nonlinear sufficient statistics $\mathbf{T}(\mathbf{r})$ is possible: $p(\mathbf{r}|x) \propto \exp(\mathbf{h}(x) \cdot \mathbf{T}(\mathbf{r}))$.

An important property of linear PPCs, central to this work, is that different projections of the population activity encode the natural parameters of the underlying posterior distribution. For example, if the posterior distribution is Gaussian (Figure 1), then $p(x|\mathbf{r}) \propto \exp\left(-\frac{1}{2}x^2\mathbf{a} \cdot \mathbf{r} + x\mathbf{b} \cdot \mathbf{r}\right)$, with $\mathbf{a} \cdot \mathbf{r}$ and $\mathbf{b} \cdot \mathbf{r}$ encoding the linear and quadratic natural parameters of the posterior. These projections are related to the expectation parameters, the mean and variance, by $\mu = \frac{\mathbf{b} \cdot \mathbf{r}}{\mathbf{a} \cdot \mathbf{r}}$ and $\sigma^2 = \frac{1}{\mathbf{a} \cdot \mathbf{r}}$.

A second important property of linear PPCs is that the variance of the encoded distribution is inversely proportional to the overall amplitude of the neural activity. Intuitively, this means that more spikes means more certainty (Figure 1).

The most fundamental probabilistic operations are the product rule and the sum rule. Linear PPCs can perform both of these operations while maintaining a consistent representation [4], a useful feature for constructing a model of canonical computation. For a log-linear probability code like linear PPCs, the product rule corresponds to weighted summation of neural activities: $p(x|\mathbf{r}_1, \mathbf{r}_2) \propto p(x|\mathbf{r}_1)p(x|\mathbf{r}_2) \iff \mathbf{r}_3 = A_1\mathbf{r}_1 + A_2\mathbf{r}_2$. In contrast, to use the sum rule to marginalize out variables, linear PPCs require nonlinear transformations of population activity. Specifically, a quadratic nonlinearity with divisive normalization performs near-optimal marginalization in linear PPCs [4]. Quadratic interactions arise naturally through coincidence detection, and divisive normalization is a nonlinear inhibitory effect widely observed in neural circuits [19, 20, 21]. Alternatively, near-optimal marginalizations on PPCs can also be performed by more general nonlinear transformations [22]. In sum, PPCs provide a biologically compatible representation of probabilistic information.

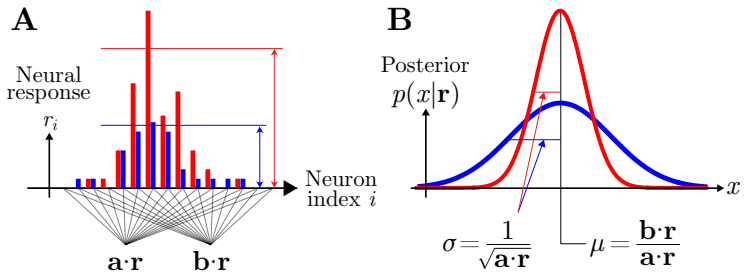

Figure 1: Key properties of linear PPCs. (**A**) Two single trial population responses for a particular stimulus, with low and high amplitudes (blue and red). The two projections $\mathbf{a}\cdot\mathbf{r}$ and $\mathbf{b}\cdot\mathbf{r}$ encode the natural parameters of the posterior. (**B**) Corresponding posteriors over stimulus variables determined by the responses in panel A. The gain or overall amplitude of the population code is inversely proportional to the variance of the posterior distribution.

3 Inference by Tree-based Reparameterization

3.1 Graphical Models

To generalize PPCs, we need to represent the joint probability distribution of many variables. A natural way to represent multivariate distributions is with probabilistic graphical models. In this work, we use the formalism of factor graphs, a type of bipartite graph in which nodes representing variables are connected to other nodes called factors representing interactions between 'cliques' or sets of variables (Figure 2A). The joint probability over all variables can then be represented as a product over cliques, $p(\boldsymbol{x}) = \frac{1}{Z}\prod_{c\in C}\psi_c(\boldsymbol{x}_c)$, where $\psi_c(\boldsymbol{x}_c)$ are nonnegative compatibility functions on the set of variables $\boldsymbol{x}_c = \{x_c | c \in C\}$ in the clique, and Z is a normalization constant. The distribution of interest will be a posterior distribution $p(\boldsymbol{x}|\mathbf{r})$ that depends on neural responses $\mathbf{r}$. Since the inference algorithm we present is unchanged with this conditioning, for notational convenience we suppress this dependence on $\mathbf{r}$.

In this paper, we focus on pairwise interactions, although our main framework generalizes naturally to richer, higher-order interactions. In a pairwise model, we allow singleton factors ψ_s for variable nodes s in a set of vertices V, and pairwise interaction factors ψ_{st} for pairs (s,t) in the set of edges E that connect those vertices. The joint distribution is then $p(\boldsymbol{x}) = \frac{1}{Z}\prod_V \psi_s(x_s)\prod_E \psi_{st}(x_s, x_t)$.

3.2 Belief Propagation and its neural plausibility

The inference problem of interest in this work is to compute the marginal distribution for each variable, $p_s(x_s) = \int_{\boldsymbol{x}\backslash x_s} p(\boldsymbol{x})\, d(\boldsymbol{x}\backslash x_s)$. This task is generally intractable. However, the factorization structure of the distribution can be used to perform inference efficiently, either exactly in the case of tree graphs, or approximately for graphs with cycles. One such algorithm is called Belief Propagation (BP) [11]. BP iteratively passes information along the graph in the form of messages $m_{st}(x_t)$ from node s to t, using only local computations that summarize the relevant aspects of other messages upstream in the graph:

$$m_{st}^{n+1}(x_t) = \int_{x_s} dx_s\, \psi_s(x_s)\psi_{st}(x_s, x_t) \prod_{u\in N(s)\backslash t} m_{us}^n(x_s) \qquad b_s(x_s) \propto \psi_s \prod_{u\in N(s)} m_{us}(x_s) \tag{2}$$

where n is the time or iteration number, and $N(s)$ is the set of neighbors of node s on the graph. The estimated marginal, called the 'belief' $b_s(x_s)$ at a node s, is proportional to the local evidence at that node $\psi_s(x_s)$ and all the messages coming into node s. Similarly, the messages themselves are determined self-consistently by combining incoming messages — except for the previous message from the target node t.

This message exclusion is critical because it prevents evidence previously passed by the target node from being counted as if it were new evidence. This exclusion only prevents overcounting on a tree graph, and is unable to prevent overcounting of evidence passed around loops. For this reason, BP is exact for trees, but only approximate for general, loopy graphs. If we use this algorithm anyway, it is called 'Loopy' Belief Propagation (LBP), and it often has quite good performance [12].

Multiple researchers have been intrigued by the possibility that the brain may perform LBP [13, 14, 5, 8, 7, 6], since it gives "a principled framework for propagating, in parallel, information and uncertainty between nodes in a network" [12]. Despite the conceptual appeal of LBP, it is important to get certain details correct: in an inference algorithm described by nonlinear dynamics, deviations from ideal behavior could in principle lead to very different outcomes.

One critically important detail is that each node must send different messages to different targets to prevent overcounting. This exclusion can render LBP neurally implausible, because neurons cannot readily send different output signals to many different target neurons. Some past work simply ignores the problem [5, 7]; the resultant overcounting destroys much of the inferential power of LBP, often performing worse than more naïve algorithms like mean-field inference. One better option is to use different readouts of population activity for different targets [6], but this approach is inefficient because it requires many readout populations for messages that differ only slightly, and requires separate optimization for each possible target. Other efforts have avoided the problem entirely by performing only unidirectional inference of low-dimensional variables that evolve over time [14]. Appealingly, one can circumvent all of these difficulties by using an alternative formulation of LBP known as Tree-based Reparameterization (TRP).

3.3 Tree-based Reparameterization

Insightful work by Wainwright, Jakkola, and Willsky [15] revealed that belief propagation can be understood as a convenient way of refactorizing a joint probability distribution, according to approximations of local marginal probabilities. For pairwise interactions, this can be written as

$$p(\boldsymbol{x}) \;=\; \frac{1}{Z}\prod_{s\in V}\psi_s(x_s)\prod_{(s,t)\in E}\psi_{st}(x_s,x_t) \;=\; \prod_{s\in V}T_s(x_s)\prod_{(s,t)\in E}\frac{T_{st}(x_s,x_t)}{T_s(x_s)T_t(x_t)} \tag{3}$$

where $T_s(x_s)$ is a so-called 'pseudomarginal' distribution of x_s and $T_{st}(x_s,x_t)$ is a joint pseudomarginal over x_s and x_t (Figure 2A–B), where T_s and T_{st} are the outcome of Loopy Belief Propagation. The name pseudomarginal comes from the fact that these quantities are always locally consistent with being marginal distributions, but they are only globally consistent with the true marginals when the graphical model is tree-structured.

These pseudomarginals can be constructed iteratively as the true marginals of a different joint distribution $p^\tau(\boldsymbol{x})$ on an isolated tree-structured subgraph τ. Compatibility functions ψ from factors remaining outside of the subgraph are collected in a residual term $r^\tau(\boldsymbol{x})$. This regrouping leaves the joint distribution unchanged:

$$p(\boldsymbol{x}) = p^\tau(\boldsymbol{x})r^\tau(\boldsymbol{x}) \tag{4}$$

The factors of p^τ are then rearranged by computing the true marginals on its subgraph τ, again preserving the joint distribution. In subsequent updates, we iteratively refactorize using the marginals of p^τ along different tree subgraphs τ (Figure 2C).

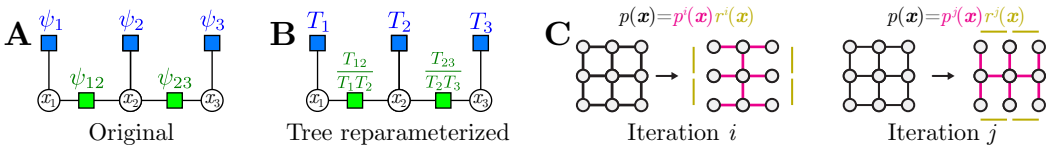

Figure 2: Visualization of tree reparameterization. (**A**) A probability distribution is specified by factors $\{\psi_s, \psi_{st}\}$ on a tree graph. (**B**) An alternative parameterization of the same distribution in terms of the marginals $\{T_s, T_{st}\}$. (**C**) Two TRP updates for a 3×3 nearest-neighbor grid of variables.

Typical LBP can be interpreted as a sequence of local reparameterizations over just two neighboring nodes and their corresponding edge [15]. Pseudomarginals are initialized at time $n = 0$ using the original factors: $T_s^0(x_s) \propto \psi_s(x_s)$ and $T_{st}^0(x_s,x_t) \propto \psi_s(x_s)\psi_t(x_t)\psi_{st}(x_s,x_t)$. At iteration $n+1$, the node and edge pseudomarginals are computed by exactly marginalizing the distribution built from previous pseudomarginals at iteration n:

$$T_s^{n+1} \propto T_s^n \prod_{u\in N(s)} \frac{1}{T_s^n}\int T_{su}^n\, dx_u \qquad\qquad T_{st}^{n+1} \propto \frac{T_{st}^n}{\left(\int T_{st}^n\, dx_t\right)\left(\int T_{st}^n\, dx_s\right)} T_s^{n+1}T_t^{n+1} \tag{5}$$

Notice that, unlike the original form of LBP, operations on graph neighborhoods $\prod_{u\in N(s)}$ do not differentiate between targets.

4 Neural implementation of TRP updates

4.1 Updating natural parameters

TRP's operation only requires updating pseudomarginals, in place, using local information. These are appealing properties for a candidate brain algorithm. This representation is also nicely compatible with the structure of PPCs: different projections of the neural activity encode the natural parameters of an exponential family distribution. It is thus useful to express the pseudomarginals and the TRP inference algorithm using vectors of sufficient statistics $\boldsymbol{\phi}_c(\boldsymbol{x}_c)$ and natural parameters $\boldsymbol{\theta}_c^n$ for each clique: $T_c^n(\boldsymbol{x}_c) = \exp\left(\boldsymbol{\theta}_c^n \cdot \boldsymbol{\phi}_c(\boldsymbol{x}_c)\right)$. For a model with at most pairwise interactions, the TRP updates (5) can be expressed in terms of these natural parameters as

$$\boldsymbol{\theta}_s^{n+1} = (1-d_s)\boldsymbol{\theta}_s^n + \sum_{u\in N(s)} \mathbf{g}_V(\boldsymbol{\theta}_{su}^n) \qquad \boldsymbol{\theta}_{st}^{n+1} = \boldsymbol{\theta}_{st}^n + Q_s\boldsymbol{\theta}_s^{n+1} + Q_t\boldsymbol{\theta}_t^{n+1} + \mathbf{g}_E(\boldsymbol{\theta}_{st}^n) \quad (6)$$

where d_s is the number of neighbors of s, and Q_s, $\mathbf{g}_V$ and $\mathbf{g}_E$ are matrices and nonlinear functions (for vertices V and edges E) that are determined by the particular graphical model (see below). Since the natural parameters reflect log-probabilities, the product rule for probabilities becomes a linear sum in $\boldsymbol{\theta}$, while the sum rule for probabilities must be implemented by nonlinear operations $\mathbf{g}$ on $\boldsymbol{\theta}$.

In the concrete case of a Gaussian graphical model, the joint distribution is given by $p(\boldsymbol{x}) \propto \exp\left(-\frac{1}{2}\boldsymbol{x}^\top\mathbf{A}\boldsymbol{x} + \mathbf{b}^\top\boldsymbol{x}\right)$, where $\mathbf{A}$ and $\mathbf{b}$ are the natural parameters, and the linear and quadratic functions $\boldsymbol{x}$ and $\boldsymbol{x}\boldsymbol{x}^\top$ are the sufficient statistics. When we reparameterize this distribution by pseudomarginals, we again have linear and quadratic sufficient statistics: two for each node, $\boldsymbol{\phi}_s = \left(-\frac{1}{2}x_s^2,\ x_s\right)^\top$, and five for each edge, $\boldsymbol{\phi}_{st} = \left(-\frac{1}{2}x_s^2,\ x_s x_t,\ -\frac{1}{2}x_t^2,\ x_s,\ x_t\right)^\top$. Each of these vectors of sufficient statistics has its own vector of natural parameters, $\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_{st}$.

To approximate the marginal probabilities, the TRP algorithm initializes the pseudomarginals to $\boldsymbol{\theta}_s^0 = \left(A_{ss},\ b_s\right)^\top$ and $\boldsymbol{\theta}_{st}^0 = \left(A_{ss},\ A_{st},\ A_{tt},\ b_s,\ b_t\right)^\top$. To update $\boldsymbol{\theta}$, we must extract the matrices Q and nonlinear functions $\mathbf{g}$ that recover the univariate marginal distribution of a bivariate gaussian T_{st}. For $T_{st}(x_s, x_t) \propto \exp\left(-\frac{1}{2}\theta_1 x_s^2 - \theta_2 x_s x_t - \frac{1}{2}\theta_3 x_t^2 + \theta_4 x_s + \theta_5 x_t\right)$, this marginal is

$$T_s(x_s) = \int dx_t\, T_{st}(x_s, x_t) \propto \exp\left(-\frac{\theta_1\theta_3 - \theta_2^2}{\theta_3}\frac{x_s^2}{2} + \frac{\theta_4\theta_3 - \theta_2\theta_5}{\theta_3}x_s\right) \quad (7)$$

Using this, we can determine the form of the weight matrices and the nonlinear functions in the TRP updates (6).

$$Q_s = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}^\top \qquad Q_t = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}^\top \quad (8)$$

$$\mathbf{g}_V(\boldsymbol{\theta}_{su}^n) = \left(\boldsymbol{\theta}_{1;su}^n - \frac{\left(\boldsymbol{\theta}_{2;su}^n\right)^2}{\boldsymbol{\theta}_{3;su}^n},\ \boldsymbol{\theta}_{4;su}^n - \frac{\boldsymbol{\theta}_{2;su}^n\boldsymbol{\theta}_{5;su}^n}{\boldsymbol{\theta}_{3;su}^n}\right)^\top \quad (9)$$

$$\mathbf{g}_E(\boldsymbol{\theta}_{st}^n) = -\left(\boldsymbol{\theta}_{1;st}^n - \frac{\left(\boldsymbol{\theta}_{2;st}^n\right)^2}{\boldsymbol{\theta}_{3;st}^n},\ 0,\ \boldsymbol{\theta}_{3;st}^n - \frac{\left(\boldsymbol{\theta}_{2;st}^n\right)^2}{\boldsymbol{\theta}_{1;st}^n},\ \boldsymbol{\theta}_{4;st}^n - \frac{\boldsymbol{\theta}_{2;st}^n\boldsymbol{\theta}_{5;st}^n}{\boldsymbol{\theta}_{3;st}^n},\ \boldsymbol{\theta}_{5;st}^n - \frac{\boldsymbol{\theta}_{2;st}^n\boldsymbol{\theta}_{4;st}^n}{\boldsymbol{\theta}_{1;st}^n}\right)^\top$$

where $\boldsymbol{\theta}_{i;st}$ is the i^{th} element of $\boldsymbol{\theta}_{st}$. Notice that these nonlinear functions are all quadratic functions with a linear divisive normalization.

4.2 Separation of Time Scales for TRP Updates

An important feature of the TRP updates is that they circumvent the 'message exclusion' problem of LBP. The TRP update for the singleton terms, (5) and (6), includes contributions from *all the neighbors* of a given node. There is no free lunch, however, and the price is that the updates at time $n+1$ depend on previous pseudomarginals at two different times, n and $n+1$. The latter update is therefore instantaneous information transmission, which is not biologically feasible.

To overcome this limitation, we observe that the brain can use fast and slow timescales $\tau_{\text{fast}} \ll \tau_{\text{slow}}$ instead of instant and delayed signals. The fast timescale would most naturally correspond to the membrane time constant of the neurons, whereas the slow timescale would emerge from network interactions. We convert the update equations to continuous time, and introduce auxiliary variables $\tilde{\boldsymbol{\theta}}$ which are lowpass-filtered versions of $\boldsymbol{\theta}$ on a slow timescale: $\tau_{\text{slow}}\dot{\tilde{\boldsymbol{\theta}}} = -\tilde{\boldsymbol{\theta}} + \boldsymbol{\theta}$. The nonlinear dynamics of (6) are then updated on a faster timescale τ_{fast} according to

$$\tau_{\text{fast}}\dot{\boldsymbol{\theta}}_s = -d_s\tilde{\boldsymbol{\theta}}_s + \sum_{u\in N(s)} \mathbf{g}_V(\tilde{\boldsymbol{\theta}}_{su}) \qquad \tau_{\text{fast}}\dot{\boldsymbol{\theta}}_{st} = Q_s\boldsymbol{\theta}_s + Q_t\boldsymbol{\theta}_t + \mathbf{g}_E(\tilde{\boldsymbol{\theta}}_{st}) \tag{10}$$

where the nonlinear terms $\mathbf{g}$ depend only on the slower, delayed activity $\tilde{\boldsymbol{\theta}}$. By concatenating these two sets of parameters, $\boldsymbol{\Theta} = (\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}})$, we obtain a coupled multidimensional dynamical system which represents the approximation to the TRP iterations:

$$\dot{\boldsymbol{\Theta}} = W\boldsymbol{\Theta} + \mathbf{G}(\boldsymbol{\Theta}) \tag{11}$$

Here the weight matrix W and the nonlinear function $\mathbf{G}$ inherit their structure from the discrete-time updates and the lowpass filtering at the fast and slow timescales.

4.3 Network Architecture

To complete our neural inference network, we now embed the nonlinear dynamics (11) into the population activity $\mathbf{r}$. Since different projections of the neural activity in a linear PPC encode natural parameters of the underlying distribution, we map neural activity onto $\boldsymbol{\Theta}$ by

$$\mathbf{r} = U\boldsymbol{\Theta} \tag{12}$$

where U is a rectangular $N_{\mathbf{r}} \times N_{\boldsymbol{\Theta}}$ embedding matrix that projects the natural parameters and their low-pass versions into the neural response space. These parameters can be decoded from the neural activity as $\boldsymbol{\Theta} = U^+\mathbf{r}$, where U^+ is the pseudoinverse of U.

Applying this basis transformation to (11), we have $\dot{\mathbf{r}} = U\dot{\boldsymbol{\Theta}} = U(W\boldsymbol{\Theta} + \mathbf{G}(\boldsymbol{\Theta})) = UWU^+\mathbf{r} + U\mathbf{G}(\mathbf{U}^+\mathbf{r})$. We then obtain the general form of the updates for the neural activity

$$\dot{\mathbf{r}} = W_L\mathbf{r} + \mathbf{G}_{NL}(\mathbf{r}) \tag{13}$$

where $W_L\mathbf{r} = UWU^+\mathbf{r}$ and $\mathbf{G}_{NL}(\mathbf{r}) = U\mathbf{G}(U^+\mathbf{r})$ correspond to the linear and nonlinear computational components that integrate and marginalize evidence, respectively. The nonlinear function on $\mathbf{r}$ inherits the structure needed for the natural parameters, such as a quadratic polynomial with a divisive normalization used in low-dimensional Gaussian marginalization problems [4], but now expanded to high-dimensional graphical models. Figure 3 depicts the network architecture for the simple graphical model from Figure 2A, both when there are distinct neural subpopulations for each factor (Figure 3A), and when the variables are fully multiplexed across the entire neural population (Figure 3B). These simple, biologically-plausible neural dynamics (13) represent a powerful, nonlinear, fully-recurrent network of PPCs which implements the TRP update equations on an underlying graphical model.

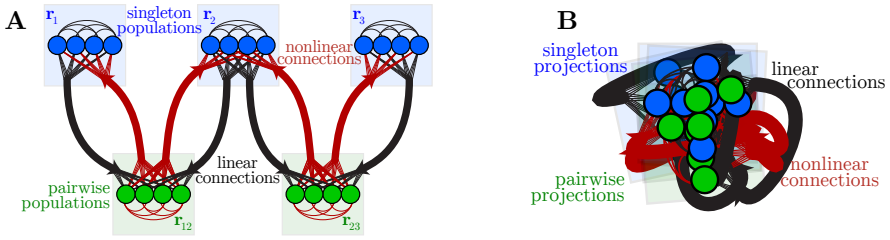

Figure 3: Distributed, nonlinear, recurrent network of neurons that performs probabilistic inference on a graphical model. (**A**) This simple case uses distinct subpopulations of neurons to represent different factors in the example model in Figure 2A. (**B**) A cartoon shows how the same distribution can be represented as distinct projections of the distributed neural activity, instead of as distinct populations. In both cases, since the neural activities encode log-probabilities, linear connections are responsible for integrating evidence while nonlinear connections perform marginalization.

5 Experiments

We evaluate the performance of our neural network on a set of small Gaussian graphical models with up to 400 interacting variables. The networks time constants were set to have a ratio of $\tau_{\text{slow}}/\tau_{\text{fast}} = 20$. Figure 4A shows the neural population dynamics as the network performs inference, along with the temporal evolution of the corresponding node and pairwise means and covariances. The neural activity exhibits a complicated timecourse, and reflects a combination of many natural parameters changing simultaneously during inference. This type of behavior is seen in neural activity recorded from behaving animals [23, 24, 25]. Figure 4B shows how the performance of the network improves with the ratio of time-scales, $\gamma \triangleq \tau_{\text{slow}}/\tau_{\text{fast}}$. The performance is quantified by the mean squared error in the inferred parameters for a given γ divided by the error for a reference $\gamma_0 = 10$.

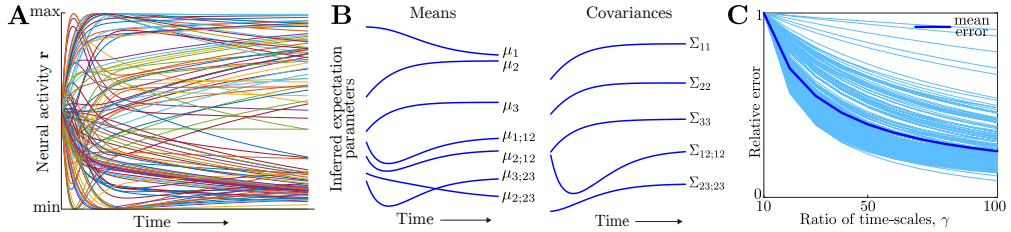

Figure 4: Dynamics of neural population activity (**A**) and the expectation parameters of the posterior distribution that the population encodes (**B**) for one trial of the tree model in Figure 2A. (**C**) Multiple simulations show that relative error decreases as a function of the ratio of fast and slow timescales γ.

Figure 5 shows that our recurrent neural network accurately infers the marginal probabilities, and reaches almost the same conclusions as loopy belief propagation. The data points are obtained from multiple simulations with different graph topologies, including graphs with many loops. Figure 6 verifies that the network is robust to noise even when there are few neurons per inferred parameter; adding more neurons improves performance since the noise can be averaged away.

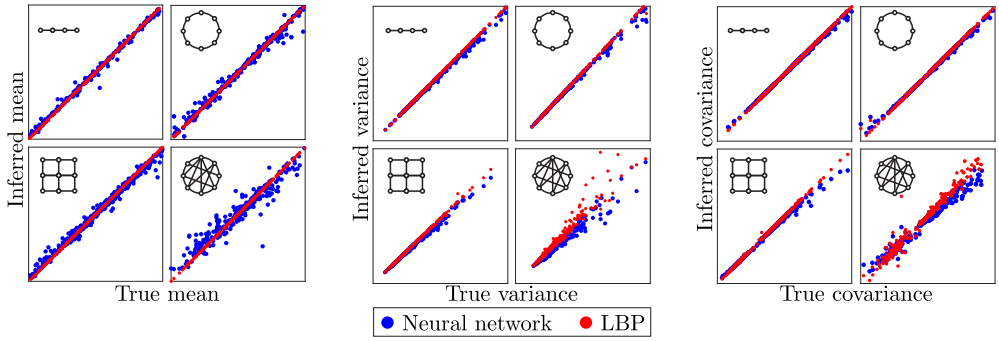

Figure 5: Inference performance of our neural network (blue) and standard loopy belief propagation (red) for a variety of graph topologies: chains, single loops, square grids up to 20×20 and densely connected graphs with up to 25 variables. The expectation parameters (means, covariances) of the pseudomarginals closely match the corresponding parameters for the true marginals.

6 Conclusion

We have shown how a biologically-plausible nonlinear recurrent network of neurons can represent a multivariate probability distribution using population codes, and can perform inference by reparameterizing the joint distribution to obtain approximate marginal probabilities.

Our network model has desirable properties beyond those lauded features of belief propagation. First, it allows for a thoroughly distributed population code, with many neurons encoding each variable and many variables encoded by each neuron. This is consistent with neural recordings in which many task-relevant features are multiplexed across a neural population [23, 24, 25], as well as with models where information is embedded in a higher-dimensional state space [26, 27].

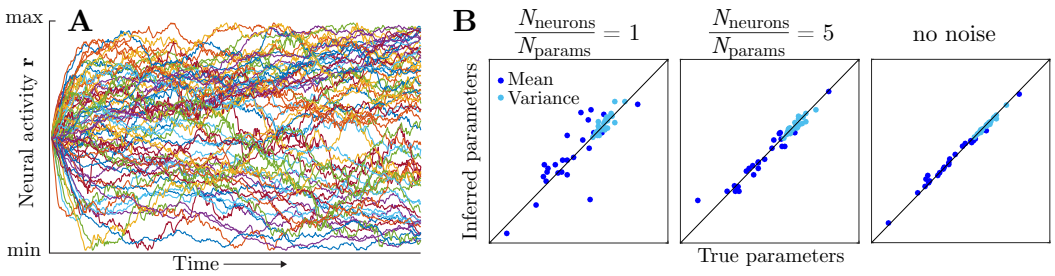

Figure 6: Network performance is robust to noise, and improves with more neurons. (**A**) Neural activity performing inference on a 5×5 square grid, in the presence of independent spatiotemporal Gaussian noise of standard deviation 0.1 times the standard deviation of each signal. (**B**) Expectation parameters (means, variances) of the node pseudomarginals closely match the corresponding parameters for the true marginals, despite the noise. Results are shown for one or five neurons per parameter in the graphical model, and for no noise (i.e. infinitely many neurons).

Second, the network performs inference in place, without using a distinct neural representation for messages, and avoids the biological implausibility associated with sending different messages about every variable to different targets. This virtue comes from exchanging multiple messages for multiple timescales. It is noteworthy that allowing two timescales prevents overcounting of evidence on loops of length two (target to source to target). This suggests a novel role of memory in static inference problems: a longer memory could be used to discount past information sent at more distant times, thus avoiding the overcounting of evidence that arises from loops of length three and greater. It may therefore be possible to develop reparameterization algorithms with all the convenient properties of LBP but with improved performance on loopy graphs.

Previous results show that the quadratic nonlinearity with divisive normalization is convenient and biologically plausible interpretable, but this precise form is not necessary: other pointwise neuronal nonlinearities are capable of producing the same high-quality marginalizations in PPCs [22]. In a distributed code, the precise nonlinear form at the neuronal scale is not important as long as the effect on the parameters is the same.

More generally, however, different nonlinear computations on the parameters implement different approximate inference algorithms. The distinct behaviors of such algorithms as mean-field inference, generalized belief propagation, and others arise from differences in their nonlinear transformations. Even Gibbs sampling can be described as a noisy nonlinear message-passing algorithm. Although LBP and its generalizations have strong appeal, we doubt the brain will use this algorithm exactly. The real nonlinear functions in the brain may implement even smarter algorithms.

To identify the brain's algorithm, it may be more revealing to measure how information is represented and transformed in a low-dimensional latent space embedded in the high-dimensional neural responses than to examine each neuronal nonlinearity in isolation. The present work is directed toward this challenge of understanding computation in this latent space. It provides a concrete example showing how distributed nonlinear computation can be distinct from localized neural computations. Learning this computation from data will be a key challenge for neuroscience. In future work we aim to recover the latent computations of our network from artificial neural recordings generated by the model. Successful model recovery would encourage us to apply these methods to large-scale neural recordings to uncover key properties of the brain's distributed nonlinear computations.

Author contributions

A2 conceived the study. A1 and A2 derived the equations. A1 implemented the computer simulations. A1 and A2 analyzed the results and wrote the paper.

Acknowledgments

A2 and A1 were supported by some grants.

References

[1] Knill DC, Richards W (1996) Perception as Bayesian inference. Cambridge University Press.

[2] Doya K (2007) Bayesian brain: Probabilistic approaches to neural coding. MIT press.

[3] Pouget A, Beck JM, Ma WJ, Latham PE (2013) Probabilistic brains: knowns and unknowns. Nature neuroscience 16: 1170–1178.

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

[5] Ott T, Stoop R (2006) The neurodynamics of belief propagation on binary markov random fields. In: Advances in neural information processing systems. pp. 1057–1064.

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

[7] Litvak S, Ullman S (2009) Cortical circuitry implementing graphical models. Neural computation 21: 3010–3056.

[8] George D, Hawkins J (2009) Towards a mathematical theory of cortical micro-circuits. PLoS Comput Biol 5: e1000532.

[9] Grabska-Barwinska A, Beck J, Pouget A, Latham P (2013) Demixing odors-fast inference in olfaction. In: Advances in Neural Information Processing Systems. pp. 1968–1976.

[10] Ma WJ, Beck JM, Latham PE, Pouget A (2006) Bayesian inference with probabilistic population codes. Nature neuroscience 9: 1432–1438.

[11] Pearl J (1988) Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann.

[12] Yedidia JS, Freeman WT, Weiss Y (2003) Understanding belief propagation and its generalizations. Exploring artificial intelligence in the new millennium 8: 236–239.

[13] Lee TS, Mumford D (2003) Hierarchical bayesian inference in the visual cortex. JOSA A 20: 1434–1448.

[14] Rao RP (2004) Hierarchical bayesian inference in networks of spiking neurons. In: Advances in neural information processing systems. pp. 1113–1120.

[15] Wainwright MJ, Jaakkola TS, Willsky AS (2003) Tree-based reparameterization framework for analysis of sum-product and related algorithms. Information Theory, IEEE Transactions on 49: 1120–1146.

[16] Jazayeri M, Movshon JA (2006) Optimal representation of sensory information by neural populations. Nature neuroscience 9: 690–696.

[17] Beck JM, Ma WJ, Kiani R, Hanks T, Churchland AK, Roitman J, Shadlen MN, et al. (2008) Probabilistic population codes for bayesian decision making. Neuron 60: 1142–1152.

[18] Graf AB, Kohn A, Jazayeri M, Movshon JA (2011) Decoding the activity of neuronal populations in macaque primary visual cortex. Nature neuroscience 14: 239–245.

[19] Heeger DJ (1992) Normalization of cell responses in cat striate cortex. Visual neuroscience 9: 181–197.

[20] Carandini M, Heeger DJ (2012) Normalization as a canonical neural computation. Nature Reviews Neuroscience 13: 51–62.

[21] Rubin DB, Van Hooser SD, Miller KD (2015) The stabilized supralinear network: A unifying circuit motif underlying multi-input integration in sensory cortex. Neuron 85: 402–417.

[22] Vasudeva Raju R, Pitkow X (2015) Marginalization in random nonlinear neural networks. In: COSYNE Abstract.

[23] Hayden BY, Platt ML (2010) Neurons in anterior cingulate cortex multiplex information about reward and action. The Journal of Neuroscience 30: 3339–3346.

[24] 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.

[25] Raposo D, Kaufman MT, Churchland AK (2014) A category-free neural population supports evolving demands during decision-making. Nature neuroscience 17: 1784–1792.

[26] Savin C, Deneve S (2014) Spatio-temporal representations of uncertainty in spiking neural networks. In: Advances in Neural Information Processing Systems. pp. 2024–2032.

[27] Archer E, Park I, Buesing L, Cunningham J, Paninski L (2015) Black box variational inference for state space models. arXiv stat.ML: 1511.07367.